1-1-2015

Community Detection Using Efficient Modularity Optimization Method: LabelMod with Single and Multi-Layer Graphs

Seokhun Bang
Purdue University

Graduate School Form 30
Updated 1/15/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Seokhun Bang

Entitled

COMMUNITY DETECTION USING EFFICIENT MODULARITY OPTIMIZATION METHOD: LABELMOD WITH SINGLE AND MULTI-LAYER GRAPHS

For the degree of Master of Science in Industrial Engineering

Is approved by the final examining committee:

Seokcheon Lee Chair	
Hyonho Chun	
Mark Lehto	

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Seokcheon Lee

Approved by: Abhijit Deshmukh 11/17/2015

Head of the Departmental Graduate Program Date

COMMUNITY DETECTION USING EFFICIENT MODULARITY OPTIMIZATION METHOD: LABELMOD WITH SINGLE AND MULTI-LAYER GRAPHS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Seokhun Bang

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Industrial Engineering

December 2015

Purdue University

West Lafayette, Indiana

To my loving parents, friends, and all the people who supported me.

ACKNOWLEDGMENTS

I would like to appreciate my major advisor, Dr. Seokcheon Lee, for his unlimited support and an opportunity to work with him. Dr. Lee was both a great teacher and a mentor while I was working with him. All of his advice were precious to me and helped me to enlarge my vision in the research. I also would like to thank my committee members, Dr. Hyonho Chun and Dr. Mark Lehto. Dr. Chun was another mentor who not only helped me in guiding a research direction but also, in understanding the materials of graph clusterings. Her devoted lessons also improved my knowledge while I was in the graduate school. Dr. Lehto gave invaluable guidance and suggestions during my thesis work.

During three years of high schools, four years of undergraduate, and two years of the graduate school in the US, my parents were the biggest supporters of me. Without their true sacrifice, I wouldn't have been able to study nine years alone in the US. My father, Dr. Dae-Wook Bang, was not only a great dad but also a respectful life mentor. My mother, Mi-Jung Park, was the one who always supported me and trusted in everything I do. I would like to sincerely appreciate them and would like to express that I love them. I also would like to acknowledge Carol Song for helping and supporting me throughout my degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABBREVIATIONS

AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion
LM	LabelMod
LPA	Label Propagation Method
MCL	Markov Cluster Algorithm
MOD	Modularity
NMI	Normalized Mutual Information
PUR	Purity
RI	Random Index
SBM	Stochastic Block Model
SC-GEN	Spectral Clustering with Generalized Eigen-Decomposition
SC-SR	Spectral Regularization
SC-ML	Spectral Clustering in Multi-layer Network
SPC	Spectral Clustering

NOMENCLATURE

A	Adjacency Matrix
C	Community Assignments
H	Entropy
I	Mutual Information
in	Inflation Parameter
L	Laplacian Matrix
P	Probability Matrix
Q	Modularity Function
r	Cut-off Parameter
s	Score Function
U	Matrix Containing the First k Eigenvectors of $u_1, ..., u_k$
W	Weighted Adjacency Matrix
Ω	Clustering Results by an Algorithm

ABSTRACT

Bang, Seokhun MSIE, Purdue University, December 2015. Community Detection Using Efficient Modularity Optimization Method: LabelMod with Single and Multi-layer Graphs. Major Professor: Seokcheon Lee.

Graph clustering is a field of study that helps reveal characteristics of communities. Systems can be viewed as networks and form communities in various areas such as biology, computer science, engineering, economics, and politics. A clustering algorithm is a tool that detects communities and it can be also considered as a pre-processing step to study the characteristics of detected communities. Many efforts were made to develop a well performing clustering algorithm in different types of networks. In recent literature, a concept of multi-layer graphs emerged, and clustering algorithms are being developed to detect communities in the multi-layer graphs. In this thesis, we propose a clustering algorithm that can be applied to both single-layer and multi-layer graphs. We test the algorithm on simulated data and real data in both single-layer and multi-layer graphs. Four performance measures were used to evaluate the performance of the proposed algorithm. We also study how the performance measures are correlated with each other and what the effects of parameter, presented in the proposed algorithm are. The thesis concludes with summary of research findings and directions of the future research.

1. INTRODUCTION

A network consists of nodes and edges. In a network, a node represents an individual identity and an edge is a connection between nodes. In most cases, systems can be viewed as networks, and this is the reason why people make efforts to analyze networks. Graph clustering is a field of study that helps to reveal characteristics of networks by identifying communities, and clustering algorithms are the tools to detect communities. Fortunato [1] also defines communities as groups of individuals that share common properties and have similar roles in the network. The objective of graph clustering is to detect community structure based on the edge connections. It does not focus on what the common properties in the same group are. Common properties are usually analyzed after communities are found. Therefore, graph clustering can be considered as a pre-processing step in understanding the characteristics of a network.

Utilizing a suitable clustering algorithm will result in the detection of distinct communities and increase the probability of features that each community has. One of the most powerful quality functions, modularity, evaluates clustering results regarding the density. The density of communities is affected by the number of between edges and within edges. Between edges are connections between nodes in different communities. Within edges connect nodes in the same community. The modularity function that was proposed by Newman et al. [2] is shown in Equation 1.1:

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij}) \delta(C_i, C_j) \tag{1.1}$$

In Equation 1.1, m is the total number of edges of the graph, A is the adjacency matrix, and P is the edge probability between node i and node j. δ is a function that has a value of 1 if node i and node j are in the same community, and 0 otherwise. In other words, the δ function detects whether node i and node j are connected with a

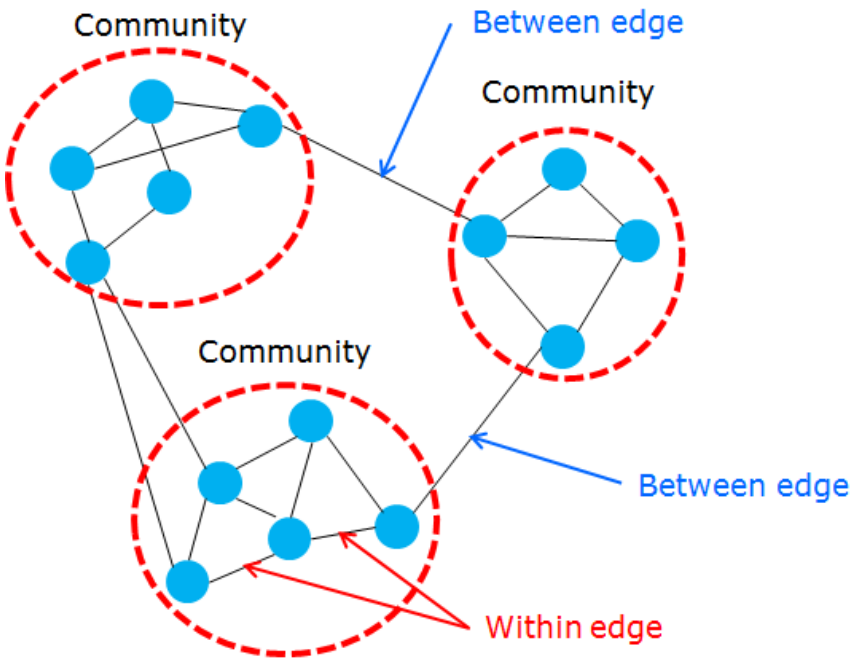


Figure 1.1. Example of Graph Clustering

between edge or a within edge. The modularity equation only sums $(A_{ij} - P_{ij})$ values with between edges. Modularity will be discussed again in Chapter 3 of this thesis.

One of the most critical assumptions in graph clustering is that the graphs are not random. The purpose of graph clustering is to detect communities with common properties. If graphs are random, nodes will not have common properties, and no relevance is expected from having community.

In this thesis, several assumptions in the course of developing community detection are made as follows:

- Adjacency matrix can be either weighted or unweighted.
- Edges are undirected.
- Every node has one and only one community assignment.

1.1 Problem Domain

There are many areas for which the associated system can be expressed as a network. Social network analysis had been already popular back in the 1930s and became one of the most important issues in sociology [3, 4]. Today, social network analysis has been expanded to social network systems such as Facebook or Twitter. These online platforms help people to communicate with others by allowing such services as messaging or photo sharing. Scholars have been researching about social groups' behaviors, and clustering algorithms were used as a pre-processing step [5]. Online retailers like Amazon already use the clusters of customers of similar interest, as a target marketing technique to recommend other items for their clients [6].

Graph clustering is also employed in the field of biology. In protein-protein interaction networks, proteins in the same community might provide similar functions within the cell [7–11]. Graph clustering can be also applied to protect from dangerous diseases that spread between humans. Figure 1.2 is an illustration of when disease occurs. Although there are several conditions that disease can transfer, disease spread by humans can be controlled by disconnecting between edges. Furthermore, a predictive model can be developed to determine which groups have higher risk of disease infection and pose the most risk of damage to a society.

1.2 Research Objectives

In graph clustering problems, the algorithms cannot always identify communities perfectly because they only consider node and edge information in the network. Moreover, most of the graph clustering problems are difficult and time-consuming to solve regarding maximizing modularity. Therefore, developing a effective and time-efficient clustering algorithm is critical.

A protein-protein interaction network is an example where the proteins' ground truth community information is unknown. In this case, accuracy cannot be measured, and instead, modularity is used. Higher modularity tends to have higher accuracy,

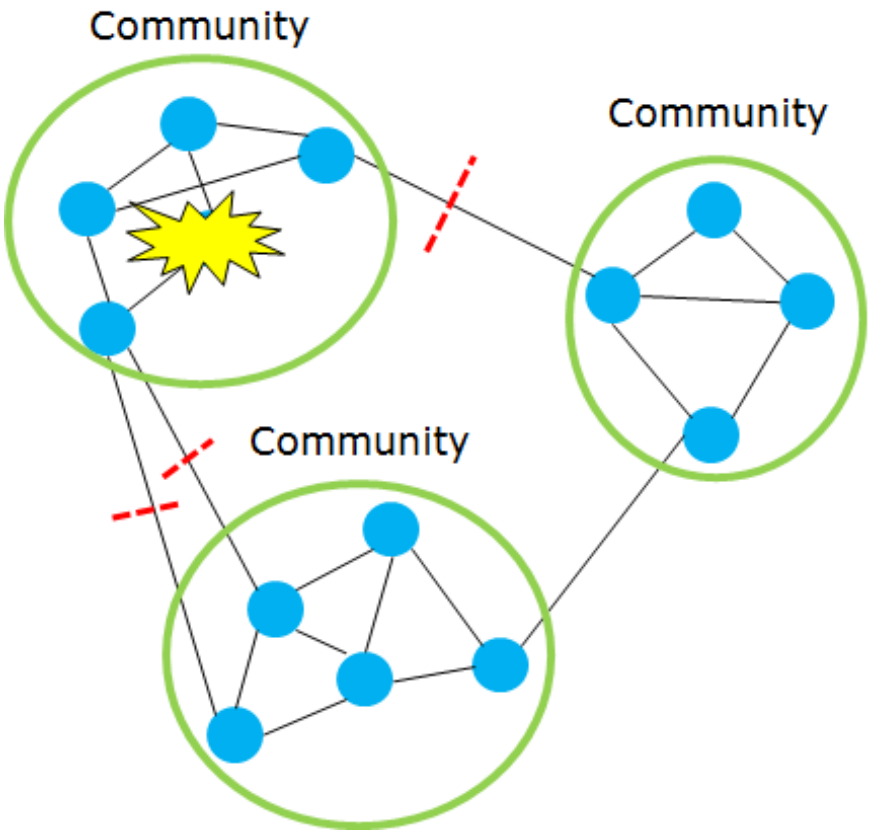


Figure 1.2. Graph Clustering Application in Disease Control

but this is not always true. There is no research about the conditions where high modularity guarantees high accuracy; the relationship between modularity and accuracy should be studied further.

Most of the currently developed algorithms are only applied to a traditional network. However, a new concept of multi-layer networks has recently emerged [12]. Multi-layer networks are graphs that have multiple slices of individual networks that have the same nodes but different edges. Like multi-layer networks, graphs can have different characteristics.

Some networks can be dense, while other networks are sparse. It will be the best practice to categorize types and find the best-performing algorithms by types of networks. However, there is no network categorization as of yet, and it is very difficult

to prove that one clustering algorithm outperforms another. Therefore, by finding a generalized clustering algorithm that guarantees a satisfactory result is beneficial. There are numerous algorithms, but the theorems and mathematical proof bases are weak [1]. By developing a new clustering algorithm, we hope that our new algorithm can contribute to the findings of new theorems and mathematical proof of graph clustering. Below, there are three main objectives of our research:

- Develop a well-performing and time-efficient clustering algorithm.
- Develop a clustering algorithm that guarantees not only high modularity but also accurate clustering results.
- Develop a generalized clustering algorithm that performs well on different types of network.

1.3 Organization of the Thesis

In Chapter 1, we briefly looked at what graph clustering is. We also discussed its applications and three main objectives of this research. In the next chapter, we look into the related literature about well-known existing algorithms, discuss a recent concept of multi-layer networks, and study several algorithms that are essential to our algorithm. In Chapter 3, we present how our methods are developed and how they perform in a single-layer network. In Chapter 4, our method is applied to multi-layer networks by simply aggregating individual single-layer networks. In the last section, Chapter 5, we conclude the thesis with our results and provide suggestions for future research.

2. LITERATURE REVIEW

Graph clustering does not have a long history compared with graph theory. The idea of graph theory first emerged in 1736 and was originated by Euler [13]. On the other hand, the idea of graph clustering became famous in the early 21st century. Newman and Girvan were the pioneers of this field who used graph clustering methods in the network [14, 15]. After the emergence of graph clustering, it is now applied to various fields like sociology [3–5], marketing [6], biology [7–11], and many more [16–18]. Even with a brief history, numerous algorithms were developed and tested. Fortunato surveyed existing algorithms and organized algorithms into seven categories [1].

In this chapter, we will survey the literature. First, various algorithms in graph clustering will be presented. We explain how algorithms work and how algorithms in several categories differ. Then, we discuss multi-layer networks, which constitute one of the types of network. An investigation about using spectral methods will follow. The chapter will conclude with a literature survey on label propagation methods.

2.1 Existing Algorithms

The first category of clustering algorithms is referred as traditional methods. Graph partitioning, hierarchical clustering, partitional clustering, and spectral clustering are known traditional methods. These algorithms were already used in different fields for different purposes. The Kernighan-Lin algorithm is one of earliest graph partitioning method that was proposed in the 1970s [19]. The algorithm was first developed to solve electric circuit partitioning. Hierarchical clustering has two approaches: The first is agglomerative algorithms where communities are created by merging each iteration. The second approach is divisive hierarchical algorithms where communities break down in each iteration. Agglomerative hierarchical algo-

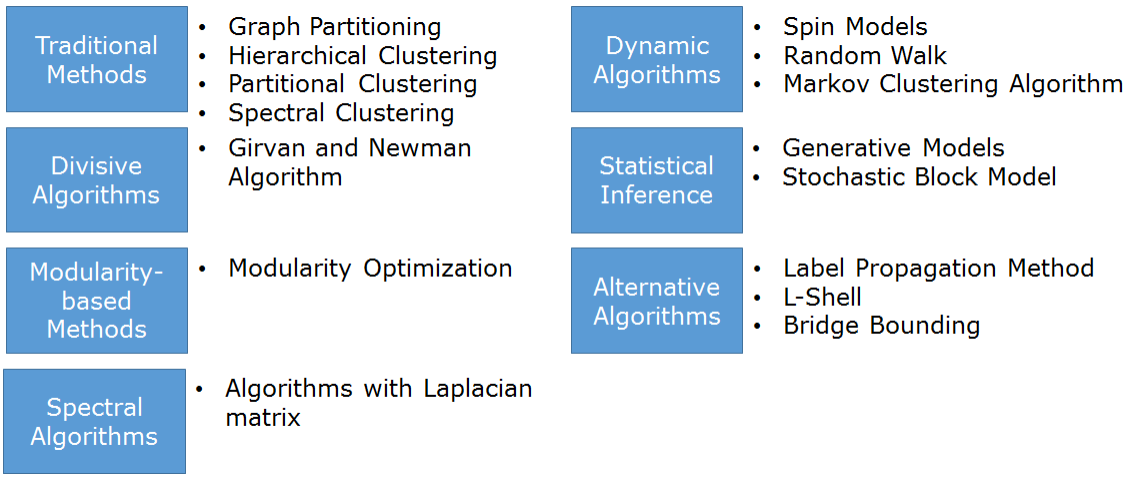


Figure 2.1. Categories of Clustering Algorithms.

rithms cluster from bottom to top while divisive hierarchical algorithms function in a top to bottom manner. In partitional clustering, k-means clustering is most often used [20]. Numerous authors have proposed an extended version of k-means clustering [21–23]. Spectral clustering is very similar to the partitional clustering method except that it uses elements of the eigenvector to derive features based on nodes and edges; then the algorithm computes a k-means clustering [24, 25]. Most of the traditional methods are limited due to their need for prior information. The number of communities should be predefined except for hierarchical clustering.

The second category is divisive algorithms. The main concept of divisive algorithms is to detect the between edges and remove them until no between edges exist. Divisive algorithms are very similar to divisive hierarchical clustering. However, divisive algorithms remove between edges rather than removing low similarity edges between pairs of nodes. The algorithm of Girvan and Newman is considered to be the start of the divisive algorithm in graph clustering [2, 14, 15]. The algorithm computes the centrality for all edges and removes the edge with the largest centrality. When there is a tie, one is randomly chosen. The algorithm iterates until it cannot find any between edge. Since the time when Girvan and Newman were the pioneers

of graph clustering, other authors have proposed variants of this algorithm [9, 23, 26]. The weakness of this algorithm is that it cannot find clusters in overlapping communities. Developing an algorithm that can deal with the overlapping communities is one of the rising research topics in this field.

The third category is modularity-based methods. Modularity is a quality function that was first proposed by Girvan and Newman, which was originally used to determine the stopping criterion of their algorithm [14]. Now, considered one of the most important methods of defining good clusterings. There are efforts to utilize this modularity function in the algorithm. In graph clustering, some algorithms that do not use a modularity function are fast but resulted in poor clusters [27–29]. Applying a modularity function has an advantage of higher probability to produce more accurate clusterings, but it may also increase the computational complexity [30–32]. Some algorithms [33–35] were developed to have both reasonable computational complexities and satisfactory accuracies.

Another category of clustering algorithms is spectral algorithms. Spectral clustering was already introduced as divisive algorithms; this category is different from the divisive algorithms but shares a similar concept. Like spectral clustering, spectral algorithms compute eigenvectors from a Laplacian matrix. However, instead of using k-means clusterings at the end, spectral algorithms use different methods. An algorithm that was proposed by Capocci et al. [36] uses eigenvector values of the right stochastic matrix where the matrix is computed from the adjacency matrix by dividing number of edges in node i.

Spin models are methods in dynamic algorithm categorization that consider a system of spins in q different states. One of the most well-known models is the Potts model [37]. Based on the Potts model, Blatt et al. [38] and Reichardt et al. [39] developed the spin model further. Another method in a dynamic algorithm is a random walk model [40]. A random walker would remain longer in a community due to the characteristics of the networks that a denser network tends to have more within edges than between edges. Zhou has developed several algorithms using the random

walk [41]. Zhou first developed an algorithm by using a distance between nodes by employing a random walk. In the next paper, Zhou measured the dissimilarity of nodes by using random walk distance [42].

Van Dogen [43] developed the Markov Cluster Algorithm (MCL). This algorithm first computes a probability matrix and it subsequently continues iterative steps of expansion and inflation until a steady state is reached. In MCL, expansion is a matrix multiplication process with the power of e. Inflation uses the following equation:

$$(\Gamma_r M)_{pq} = (M_{pq})^r / \sum_{i=1}^{k} (M_{iq})^r \tag{2.1}$$

Here, M is a stochastic matrix such that a value in row i and column j is the probability of going from node j to node i. There is an inflation parameter, r, which is always greater than 1. An inflation operator works like a normalization process in that if the nodes have similarity regarding the random walk distance, the value in the matrix will increase. In the opposite case, if the nodes have dissimilarity, the value will decrease. MCL and its inflation operator will be discussed again later in Chapter 3.

Statistical inference is another category in which the method starts with a set of observations and decides a hypothesis of a model. Two famous models in statistical inference methods are generative models and block modeling. A generative model uses Bayes' theorem and it has an aim to find a parameter $\{\theta\}$ that maximizes the posterior distribution below:

$$P(\{\theta\}|D) = \frac{P(D|\{\theta\})P(\{\theta\})}{\int P(D|\{\theta\})P(\{\theta\})d\theta} \tag{2.2}$$

Here, $P(\{\theta\})$ is the prior distribution and D is the information about the system. The problem of the generative model and of using Bayesian inference is the fact that the equation includes the integral. Having an integral in the equation will result in more computational complexity. Also, the choice of the prior distribution is not clear as well. Some researchers tried to apply this model in social networks [44–46] and biological networks [47, 48].

Another well-known statistical approach is block modeling. Block modeling finds a group of nodes with similar characteristics. Nodes form a community by two types of equivalence: structural equivalence [49] where nodes have same neighbors and regular equivalence [50] where nodes have similar patterns of connection. As an extension, the Stochastic Block Model [51] uses a stochastic equivalence that is analogous to the structural equivalence where similar linking probabilities are considered to be of the same class. Because a statistical inference method is model-based rather than algorithm-based, it is important to select a model that suits the data well. Common statistical model selection heuristics include the Akaike Information Criterion (AIC) [52] and the Bayesian Information Criterion (BIC) [53].

Sometimes, communities are detected by using the information theory. However, a deeper review of information theory is out of our research scope. Mutual information [54] is a measure that informs about how much one solution is learned when the other solution is known. Mutual information can be computed by subtracting a conditional entropy of two clustering results from a marginal entropy of one clustering result. The equation for mutual information is as follows [1]:

$$\begin{aligned} I(X,Y) &= H(X) - H(X|Y) \\ &= \sum_{x} P(x)\ log\frac{1}{P(x)} - \sum_{xy} P(x,y)\ log\frac{1}{P(x|y)} \\ &= \sum_{x}\sum_{y} P(x,y)\ log\frac{P(x,y)}{P(x)P(y)} \end{aligned} \tag{2.3}$$

In the equation, $H(X)$ is the marginal entropy of clustering assignment X and $H(X|Y)$ is the conditional entropy of X and Y. Both marginal entropy and conditional entropy can be computed by utilizing marginal probability of x, joint probability of x and y, and conditional probability of x and y.

The last category, alternative algorithms, includes all of the algorithms that are not classified in previous categories. Methods like L-shell [55], Bridge Bounding [56] and other heuristic algorithms are in this category. The Label Propagation method

[57] is also an alternative algorithm. However, it will be discussed in more detail in Section 2.4.

2.2 Multi-layer Networks

Mucha et al. developed a new framework of community structure named multi-slice networks [12]. By its definition, slices of networks can represent variations over time, or different connections at various scales. Figure 2.2 is an example of multi-slice networks. In each slice, the nodes are the same. However, the edges connecting the nodes are different across the slices. Mucha et al. used 110 slices of vote similarities [12]. They also used four slices of Facebook friendships, picture friendships, roommates, and housing groups on the "Tastes, Ties, and Time" network [58].

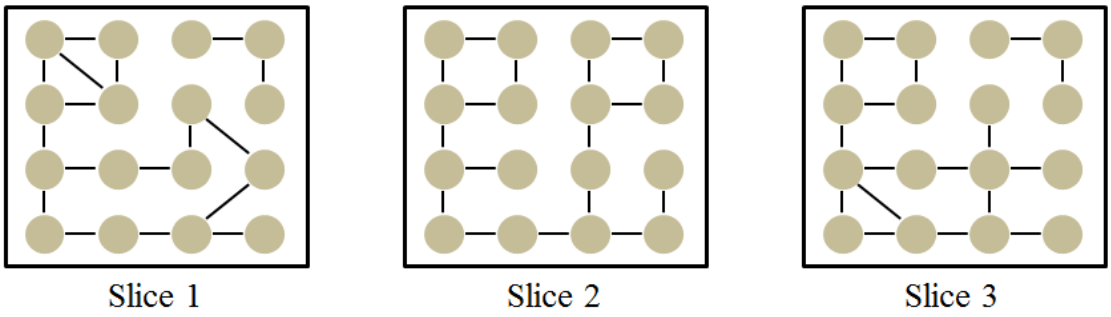


Figure 2.2. Example of Multislice Networks.

After the emergence of the multislice concept, there were several efforts to apply clustering algorithms in multi-layers of graphs. Dong et al. first developed two methods, clustering with generalized eigen-decomposition (**SC-GED**) and spectral regularization (**SC-SR**) [4]. Then Dong extended their algorithm (**SC-ML**) by using the modified Laplacian matrix L_{mod} [10]. SC-ML outperformed its competing algorithms in purity, normalized mutual information, and random index. In addition to Dong's work, Zhang et al. used the multi-layer graph concept in gene co-expression

networks [59]. Hu et al. also applied the multislice concept in optimizing modularity [60]. In their study, clustering results were used as an image segmentation. The pixels in the image functioned as an adjacency matrix. However, due to a large number of pixels, it required a considerable amount of memory and had an expensive computational time. The authors also pointed out in their paper that a computational improvement is required to deal with an increased amount of network data.

2.3 Spectral Clustering

As briefly explained in the previous section, spectral clustering is an algorithm that uses an eigenvector decomposition of a Laplacian matrix. Then the algorithm runs a k-means clustering algorithm by using few major eigenvectors. The more detailed procedure is shown in Figure 2.3. By combining a spectral method with k-means clustering, it became one of the most well-known algorithms for graph clustering. Some researchers used a spectral clustering in multi-layer graphs [4, 10, 59]. However, selecting an appropriate k value is another problem in spectral clustering. When ground truth communities are known, the algorithm can use the number of ground truth communities as k. On the other hand when ground truth communities are not known, the selection of k is vague. Pham et al. provided a guideline on how k should be selected in k-means clustering [61]. In our research, spectral clustering is one of the algorithms that will be compared with our proposed algorithm.

Algorithm **Spectral Clustering**

1: **Input:**
W: $n \times n$ weighted adjacency matrix.
k: Target number of clusters.

2: Compute the degree matrix D and the normalized graph Laplacian matrix.
$L = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$

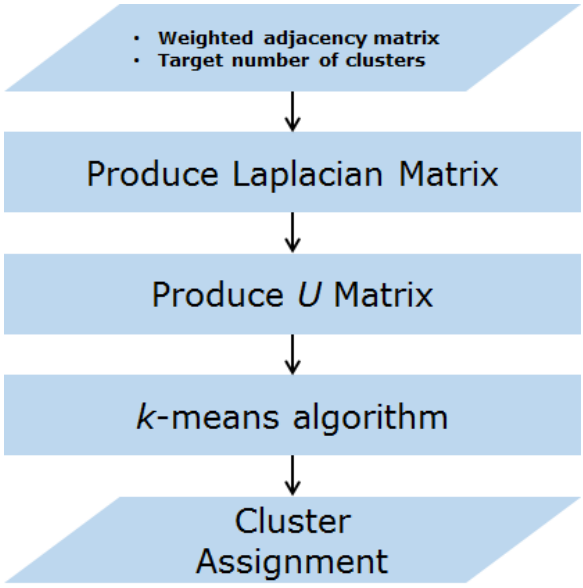
3: Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the first k eigenvectors $u_1, \ldots, u_k$ of L.

4: Let $y_i \in \mathbb{R}^k$ $(j = 1, \ldots, n)$ be the j-th row of U.

5: Cluster $y_i \in \mathbb{R}^k$ into k clusters $C_1, \ldots, C_k$ using the k-means algorithm.

6: **Output:**
$C_1, \ldots, C_k$: The cluster assignment.

(a) Pseudo Codes.



(b) Flowchart.

Figure 2.3. Spectral Clustering Algorithm.

2.4 Label Propagation Method

The Label Propagation Method (LPA) was first proposed by Raghavan et al. to be used in a large-scale network graph clustering [57]. The algorithm starts with independent labels. At every step, nodes look for their neighbors' labels and change their label with the maximum occurring labels. When there are ties among neighbors, labels are assigned uniformly and randomly from the neighbors. The algorithm continues the iterative process until there are no label changes in the network. After finishing this iterative process, nodes with same labels are considered to be in the same community.

The algorithm was first proposed in a non-overlapping community situation where nodes can have only one community assignment. However, more research was conducted to solve a clustering problem in overlapping communities [62–64] Xie et al. improved both computational efficiency and the quality of clustering by proposing a new update rule and label propagation criterion [65]. Figure 2.4 present a detailed procedure of how the algorithms work.

Algorithm Label Propagation Method

1: **Input:**
A: $n \times n$ adjacency matrix.

2: Initialize the labels at nodes in the network.
For a given node x, $C_x(t=0) = x$.

3: Set $t = 1$.

Repeat

4: Arrange the nodes in the network in a random order and set it to X.

5: For each $x \in X$ chosen in that specific order,
let $C_x(t) = f\left(C_{x_{t1}}(t), \ldots, C_{x_{tm}}(t), C_{x_{t(m+1)}}(t-1), \ldots, C_{x_{tk}}(t-1)\right)$.
f returns the label occurring with the highest frequency among neighbors and ties broken uniformly randomly.

6: $t = t + 1$.

7: **Until:** Every node has a label that the maximum number of their neighbors have.

8: **Output:**
$C_1, \ldots, C_k$: The cluster assignment.

Figure 2.4. Pseudo Codes for Label Propagation Methods.

Despite the fact that LPA has a significant advantage of performing in a near-linear time to run the algorithm, LPA has several disadvantages. First, LPA has a convergence issue. The network that is bipartite or a nearly bipartite will cause oscillations of labels and will not form clusters. To deal with this challenge, Raghavan et al. suggests using an asynchronous update instead of a synchronous update. In a synchronous update, the algorithm only considers previous iterations labels of a nodes neighbors. On the other hand in an asynchronous update, the algorithm considers neighbors labels that were updated in both current and previous iterations. By using an asynchronous update, the algorithm can have a wider range of labels and can use both past and present information of the labels.

The second disadvantage and the most critical drawback is the randomness issue. Because this algorithm uses an asynchronous update, the sequence of the update differs by runs and the tie breaker using uniform random selection aggravates the diffusion of results. Raghavan et al. suggested using an aggregation of solutions [57]. However, Tibèly et al. showed that the aggregation method would fragment the communities into several pieces [66]. Even with an effort to resolve the randomness of LPA solutions, there has been no satisfactory extension to date.

3. LABELMOD IN SINGLE-LAYER GRAPHS

In this chapter, we will discuss LPA further as well as its inherited algorithm LabelRank. First, the LabelRank algorithm will be explained because our proposed algorithm follows similar steps. Then, the proposed algorithm will be described in more details. Later in this chapter, the proposed algorithm will be applied to both simulated data and real data in a single-layer graph. The proposed algorithm's performance will be compared with other algorithms.

3.1 LabelRank Algorithm

In Chapter 2, the literature survey described what LPA is. In summary, LPA is a heuristic algorithm that propagates a node's label to its neighbors. Because it does not utilize matrix operation in the algorithm, the algorithm is fast and efficient. However, to solve the label's oscillation problem, the algorithm randomly starts at each iteration and updates asynchronously. Because of the algorithm's characteristics, LPA produces different clustering results. Xie et al. have proposed an algorithm named LabelRank to resolve this kind of random clustering result that characterizes LPA [67].

LabelRank is similar to the Markov Cluster Algorithm (MCL) while it keeps the properties of LPA. MCL is based on the simulation of random walks that use two operators, expansion, and inflation. Two operators continue until there are no changes in the probability matrix [68]. LabelRank resembles MCL in using an inflation operator. Also, both algorithms pursue the process of controlling and normalizing a matrix. However, the most significant difference between the two algorithms is that LabelRank utilizes the label distribution matrix that normalizes after each iteration while MCL uses probability matrix to normalize. LabelRank retains the advantages

of fast and efficient characteristics of the LPA. The modularity of LabelRank was at least similar or even better than its predecessor algorithms, LPA, and MCL. However, there were questionable parts of the algorithm that needed to be improved.

The algorithm that we are proposing is similar to the LabelRank algorithm by Xie et al. The main idea of LabelRank is to perform iterative steps of propagation and normalization in the network. Although there are two key steps, the algorithm consists of four operators: 1) propagation, 2) inflation, 3) cut-off, and 4) conditional update. LabelRank maintains different labels throughout the process and nodes with the highest probability label at the end of the algorithm form a community. Since all processes are operations of a matrix, there is no randomness issue. Indeed, the algorithm is synchronous, and it does not have the problem of label oscillations even in a bipartite network. The following are detailed descriptions of the algorithm:

1) **Propagation**: With a given adjacency matrix of A ($n \times n$ matrix whose entries are 1 if there are edges connected between nodes and 0 otherwise), it is possible to compute a label distribution matrix of P ($n \times n$ matrix that assigns equal probability on nodes that have edges). The label distribution matrix can be obtained by the following equation:

$$P_{ij} = \frac{1}{k_i}, \qquad \forall j \quad s.t.\ A_{ij} = 1 \tag{3.1}$$

Once the label distribution matrix is created, it can be updated by matrix multiplication with the adjacency matrix. The propagation operator can be applied by the following equation:

$$P^{'} = A \times P \tag{3.2}$$

2) **Inflation**: The inflation operator concept originally came from the MCL. In MCL, the inflation operator works for the stochastic matrix while the label distribution matrix is applied in LabelRank [67].

$$P'' = \Gamma_{in} P' = (P')^{in} / \sum_{j=1}^{n} (P')^{in} \tag{3.3}$$

There is one parameter, in, that controls the inflation. If two nodes are strongly connected, the operator will increase the probability. On the other hand, the operator will decrease the probability if two nodes are weakly connected. After iteration, the value in the matrix will likely be close to 0 or 1.

3) **Cut-off**: After the inflation operator, there could be small probabilities in the label distribution matrix. These low probabilities will eventually merge to zero after iteration. Both the inflation operator and the cut-off operator are key hard thresholding steps of the algorithm.

4) **Explicit Conditional Update**: Xie et al. mentioned that well-clustered communities can be detected far before the convergence [67]. Therefore a conditional update operator helps to find the best solution before the convergence. The node gets updated only when there are significant differences. A detailed update condition is the following:

$$\sum_{j \in Nb(i)} isSubset(C_i^*, C_j^*) \leq q k_i \tag{3.4}$$

Here, C_i^* is the maximum label set with the maximum probability at node i at the previous iteration and the function $isSubset$ compares with its neighbors maximum labels. If $C_i^* \subseteq C_j^*$, the function returns 1 and it returns 0 if C_i^* is not a subset of C_j^*. There are parameter $q \in [0, 1]$ and k_i the degree of node i to make a condition whether or not to change the labels.

The algorithm continues iterative steps of four operators until there are no changes in the node labels. Once the iterative steps are completed, the algorithm detects a community membership of each node with a certain probability. The most significant

strength of this algorithm is that it keeps the concepts of both LPA and MCL while the randomness issue is eliminated.

Algorithm LabelRank

1: Add self-loop to adjacency matrix A.

2: Initialize the label distribution P using $P_{ij} = \frac{1}{k_j}, \forall\, j\ s.t.\ A_{ij} = 1.$

3: **Repeat**

Propagation Operator: $P' = A \times P$

Inflation Operator: $P_i'' = \Gamma_{in} P_i' = (P_i')^{in} / \sum_{j=1}^{n} (P_i')^{in}$

Cutoff Operator: Assign any value that is less than the cutoff parameter, r, to 0.

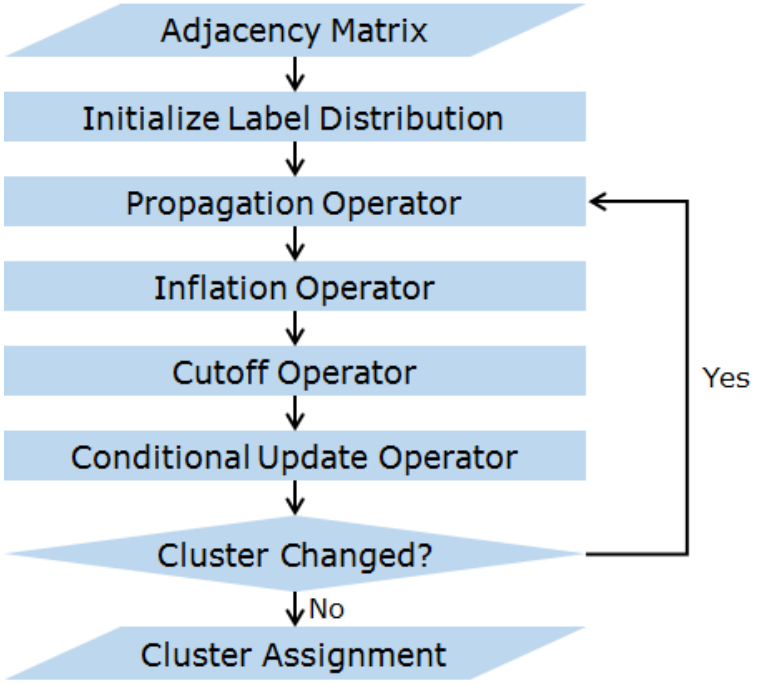
Conditional Update Operator: $\sum_{j \in Nb(i)} isSubset(C_i^*, C_j^*) \leq qk_i$

7: **Until:** There is no change in cluster assignment.

8: **Output:**

$C_1, \ldots, C_k$: The cluster assignment.

(a) Pseudo Codes.

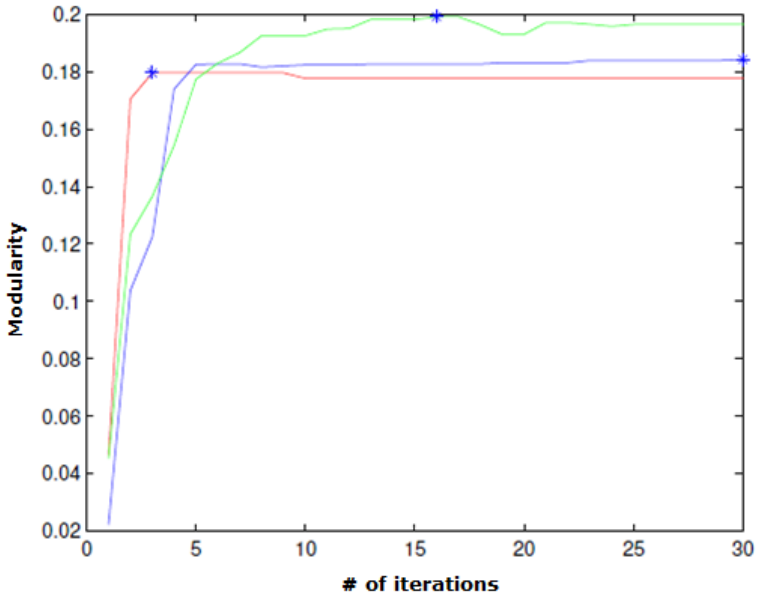


(b) Flowchart.

Figure 3.1. LabelRank Algorithm.

3.2 Algorithm Development

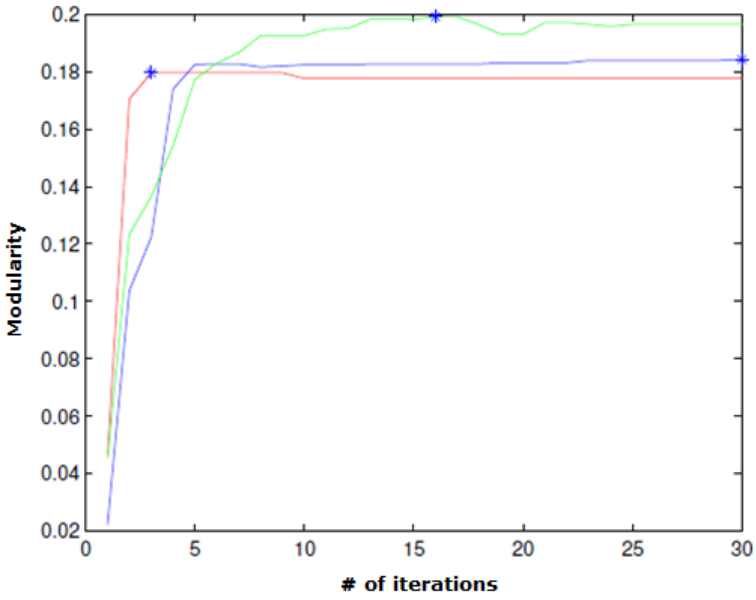
In the conditional update operator of the original LabelRank algorithm, a node's label changes whenever the conditional update statement is satisfied. However, when the algorithm was tested, we found that the conditional update will be greatly dependent on the parameter q. None of the literature papers mention this conditional update, and the reason for utilizing a conditional update operator was not sufficient. However, we found that the previous three operators are easy to implement and have fast performance. Therefore, we analyzed the LabelRank algorithm to determine its significant characteristics and developed an algorithm by introducing a new conditional update and stopping criterion.



Different color indicates different networks.

Figure 3.2. Modularity Change by Iterations in Real Graphs.

Figures 3.2 and 3.3 are plots of modularity changes over 30 iterations. A blue asterisk (*) is a maximum modularity among 30 iterations. A modularity reaches the maximum and then stabilizes in Figure 3.2. Often, modularity oscillates, but the oscillation stabilizes within five iterations. On the other hand, a modularity



Different color indicates different runs.

Figure 3.3. Modularity Change by Iterations in Simulated Graphs.

value reaches its maximum and suddenly decreases in Figure 3.3. By inspecting both figures, modularity reaches its maximum and then its value will either decrease or stabilize.

Our proposed algorithm, LabelMod, utilizes the fact that a modularity either decreases or stabilizes when the iteration increases in LabelRank. LabelMod keeps propagation, inflation, and cut-off operators and includes the modularity operator instead of the conditional update operator. The modularity operator works in each iteration and evaluates whether the algorithm should stop or continue to the next iteration. Among several equations of modularity proposed by different authors, we use the following equation [69]:

$$Q = \frac{1}{4m} Tr(S^T B S) \tag{3.5}$$

In this equation, m is the total number of edges in the network. S is the non-square matrix that has a value of 1 if the nodes belong to the same group, and 0 otherwise.

B is the modularity matrix where $B = A - \frac{k_i k_j}{2m}$. LabelMod's modularity operator compares previous modularity with current modularity to check if it decreased. If modularity values are having a stabilizing or decreasing pattern for five iterations, the algorithm stops and selects a maximum modularity among iterations.

This stopping criterion is intuitive but powerful. It does not guarantee a global maximum modularity value. However, it guarantees a local maximum modularity value. Mathematical programming techniques would find a global maximum value, but then, the computational cost would be very expensive and cannot be considered as a suitable algorithm. There is always a tradeoff in which accuracy requires more time to be spent. LabelMod is an algorithm that balance both accuracy and computational time. Because of its balanced nature, it can be applied to any type of network even with no priori information.

Algorithm LabelMod

1: Add self-loop to adjacency matrix A.

2: Initialize the label distribution P using $P_{ij} = \frac{1}{k_j}, \forall\, j\ s.t.\ A_{ij} = 1.$

3: **Repeat**

Propagation Operator: $P' = A \times P$

Inflation Operator: $P_i'' = \Gamma_{in} P_i' = (P_i')^{in} / \sum_{j=1}^{n} (P_i')^{in}$

Cutoff Operator: Assign any value that is less than the cutoff parameter, r, to 0.

Modularity Operator: $Q = \frac{1}{4m} Tr(S^T B S)$

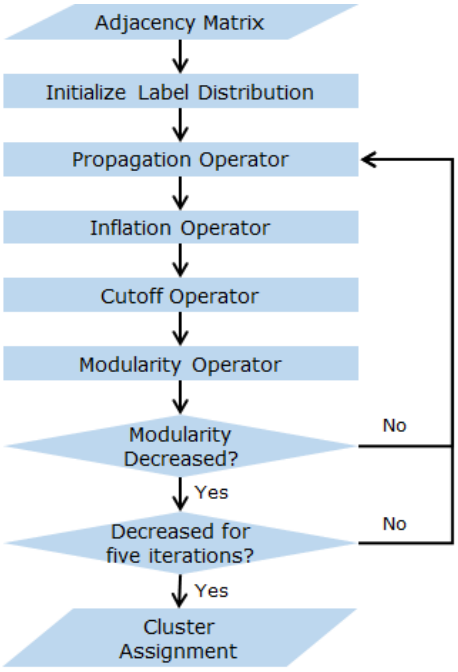
7: **Until:** The modularity of five consecutive iterations are equal or showing a decreasing trend.

8: Among all iterations, choose an iteration that has the maximum modularity.

9: **Output:**

$C_1, \ldots, C_k$: The cluster assignment.

(a) Pseudo Codes.



(b) Flowchart.

Figure 3.4. LabelMod Algorithm.

3.3 Results

In our research, LabelMod algorithm was applied to 1) simulated graphs, and 2) real graphs. In this section, LabelMod's performance will be compared with Spectral Clustering (SPC) which is one of the most well-known algorithms, and the Label Propagation (LPA) Method. For the algorithms' performance comparison, random index (RI) [70], purity (PUR), normalized mutual information (NMI) [71], and modularity were measured. For both the simulated graphs and the real graphs, ground truth communities were given.

The first performance measure, random index, is an accuracy measure based on a binary decision. After communities are detected, random index compares each node's community assignment with the ground truth community.

$$RI(\Omega, C) = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.6}$$

Here, TP, TN, FP, and FN stand for true positive, true negative, false positive, and false negative. Ω represent communities found by the algorithm and C represent ground truth communities. By summing true positive and true negative values, the equation counts how many nodes are correctly assigned to the same community and correctly assigned to different communities. The counted value then is divided by the sum of total possibilities. Random index values range from 0 to 1 and are expressed in decimals.

Purity is another performance measure. It finds a maximum intersection between the algorithm's community and ground truth community. Then it sums all of the algorithm's communities and divides by the total number of nodes. The equation for the purity is as follows:

$$Purity(\Omega, C) = \frac{1}{N} \sum_{k} max_j |\omega_k \cap C_j| \tag{3.7}$$

Purity can be also considered as one of the accuracy measures, but, it also has one significant problem. Even in the case where the algorithm produces more communities than the ground truth communities, purity can be 1. Smaller communities

can be aggregated together to form a ground truth community. Purity is not a useful measure if one wants to find an exact clustering with equal community sizes as the ground truth. However, purity is useful in clustering in overlapping communities or hierarchical structures.

The last performance measure is normalized mutual information. NMI uses mutual information between the algorithm community and ground truth community. Then it is normalized using the entropy of clusters. There are many different ways to normalize mutual information; we used the following equation [72]:

$$NMI(\Omega, C) = \frac{I(\Omega; C)}{[H(\Omega) + H(C)]/2} \tag{3.8}$$

Mutual information measures the information that two clustering results share. Here, $I(\Omega; C)$ refers to the mutual information and $H(\Omega)$ or $H(C)$ refer to the entropy. Mutual information and entropy can be computed by using the equations below:

$$H(\Omega) = -\sum_{i=1}^{R} \frac{a_i}{N} log \frac{a_i}{N} \tag{3.9}$$

$$I(\Omega; C) = \sum_{i=1}^{R} \sum_{j=1}^{C} \frac{n_{ij}}{N} log \frac{n_{ij}/N}{a_i b_j / N^2} \tag{3.10}$$

N is the total number of nodes, and n_{ij} is the number of nodes that are common between the algorithm's community and the ground truth community. a_i is $\sum_i n_{ij}$ and b_j is $\sum_j n_{ij}$.

By using three performance measures together, three algorithms can be compared objectively. Along with three performance measures, modularity was calculated for the three algorithms as well. Since LabelMod has a modularity operator inside its algorithm, it is expected that LabelMod will have higher modularity most of the time.

3.3.1 Graph Simulation

In graph clustering, the Stochastic Block Model (SBM) can generate a random network by assigning each node with a ground truth community [73]. First, a probability

matrix that will be used to construct the adjacency matrix is needed. A probability matrix is an $n \times n$ matrix where n is the total number of communities. Diagonal entries in the matrix are within community edge probability, and the rest are between community edge probability. An adjacency matrix can be produced by utilizing the probability matrix. Zhang et al. also used SBM to produce three communities with a given probability matrix as follows [59]:

$$P_1 = \frac{1}{n}\begin{pmatrix} 16 & 0 & 0 \\ 0 & 18 & 0 \\ 0 & 0 & 17 \end{pmatrix}, \quad P_2 = \frac{1}{n}\begin{pmatrix} 16 & 0.4 & 0.6 \\ 0.4 & 18 & 0.55 \\ 0.6 & 0.55 & 17 \end{pmatrix},$$

$$P_3 = \frac{1}{n}\begin{pmatrix} 16 & 0.8 & 1.2 \\ 0.8 & 18 & 1.1 \\ 1.2 & 1.1 & 17 \end{pmatrix}, \quad P_4 = \frac{1}{n}\begin{pmatrix} 16 & 1.2 & 1.8 \\ 1.2 & 18 & 1.65 \\ 1.8 & 1.65 & 17 \end{pmatrix}$$

Figure 3.5. Probability Matrix Used in Simulation Graphs

In their experiment, they had four different probability matrices with 50 runs and three communities in each run. This task was done with two different conditions; in one condition, an equal community size was used, and the other had a different community size. The same experimental method was conducted to test the performance of LabelMod. When the community size was the same, the network had 50 nodes in each one community, and the network consisted of three communities. When the community size was different, the network had 30 nodes in two communities and 90 nodes in one. In simulated studies, we found that different community size networks have slightly lower performance measures with higher standard deviations. Table 3.1 includes random index, purity, normalized mutual information, modularity and performance time data with the same community size for four conditions and Table 3.2 includes the same performance measures but with different community size.

When the community size was the same, LabelMod produced high-performance measures. Compared with other algorithms it especially performed better in normal-

ized mutual information. Also, as expected, LabelMod's modularity outperformed the other two algorithms' modularity. However, the performance time was relatively higher than the other algorithms. The spectral clustering algorithm outperformed with respect to performance times; this was because it does not require many computations to calculate a Laplacian matrix. Nevertheless, it uses k-means clustering that is one of the most popular algorithms in the field. Although spectral clustering was the fastest, it did not always produced a satisfactory result. In the P_1 matrix, there is no between edge probability. In this case, spectral clustering failed many times. When no between community edge exists, an eigenvector decomposition produces an empty Laplacian matrix that causes a failure in k-means clustering.

When the community size was different, the performance measure continued to decrease as more between community edges were produced. LabelMod still had the highest modularity at all times, but the spectral clustering maintained its performance. Spectral clustering's performance time was the fastest and LabelMod was the slowest-performing algorithm in the experiment. However, when the network had more edges, the algorithm performed faster. The performance time is highly related to the total number of nodes rather than the total number of edges. Most of the computational time is spent in matrix operations. If the network has more nodes, the size of the matrix also increases and causes an expensive computational cost. On the other hand, when there are more edges in the network, edges interact more actively in the inflation step.

Table 3.1: Simulated Graphs Result (Same Community Size)

	P_1			P_2			P_3			P_4		
	SPC	**LPA**	**LM**	**SPC**	**LPA**	**LM**	**SPC**	**LPA**	**LM**	**SPC**	**LPA**	**LM**
RI	0.92 (0.12)	0.85 (0.07)	0.99 (0.01)	0.92 (0.14)	0.83 (0.07)	0.99 (0.02)	0.91 (0.14)	0.83 (0.06)	0.97 (0.04)	0.91 (0.14)	0.82 (0.11)	0.93 (0.08)
PUR	0.89 (0.16)	1 (0)	1 (0)	0.89 (0.16)	0.98 (0.01)	0.99 (0)	0.88 (0.17)	0.96 (0.03)	0.98 (0.02)	0.88 (0.17)	0.91 (0.12)	0.95 (0.05)
NMI	0.89 (0.15)	0.71 (0.10)	0.98 (0.03)	0.87 (0.18)	0.66 (0.10)	0.96 (0.05)	0.84 (0.19)	0.62 (0.11)	0.89 (0.11)	0.81 (0.21)	0.61 (0.16)	0.80 (0.16)
Modularity	0.2492 (0.04)	0.2579 (0.23)	0.3071 (0)	0.2336 (0.04)	0.2358 (0.03)	0.2859 (0)	0.2167 (0.04)	0.2235 (0.02)	0.2627 (0.01)	0.2045 (0.04)	0.2082 (0.03)	0.2399 (0.02)
Time	0.06s (0.02s)	1.01s (0.34s)	1.04s (0.22s)	0.08s (0.02s)	0.94s (0.29s)	1.06s (0.16s)	0.08s (0.02s)	1.03s (0.25s)	1.04s (0.13s)	0.09s (0.03s)	1.14s (0.36s)	1.15s (0.25s)

SPC: Spectral Clustering / LPA: Label Propagation Method / LM: LabelMod

Table 3.2: Simulated Graphs Result (Different Community Size)

	P_1			P_2			P_3			P_4		
	SPC	**LPA**	**LM**	**SPC**	**LPA**	**LM**	**SPC**	**LPA**	**LM**	**SPC**	**LPA**	**LM**
RI	0.88 (0.14)	0.90 (0.02)	0.94 (0.06)	0.83 (0.17)	0.89 (0.05)	0.88 (0.05)	0.89 (0.10)	0.76 (0.15)	0.86 (0.04)	0.89 (0.11)	0.55 (0.19)	0.85 (0.04)
PUR	0.89 (0.10)	1 (0)	1 (0)	0.83 (0.11)	0.98 (0.01)	0.98 (0.1)	0.85 (0.10)	0.91 (0.07)	0.97 (0.01)	0.86 (0.08)	0.78 (0.11)	0.96 (0.02)
NMI	0.83 (0.17)	0.74 (0.02)	0.84(0.11)	0.63 (0.24)	0.69 (0.05)	0.69 (0.09)	0.69 (0.16)	0.56 (0.11)	0.63 (0.06)	0.67 (0.15)	0.37 (0.16)	0.59 (0.06)
Modularity	0.1574 (0.03)	0.1590 (0.01)	0.1766 (0)	0.1354 (0.03)	0.1518 (0.01)	0.1618 (0.01)	0.1400 (0.02)	0.1291 (0.02)	0.1516 (0.01)	0.1414 (0.02)	0.0900 (0.03)	0.1479 (0.01)
Time	0.07s (0.02s)	0.77s (0.20s)	1.15s (0.30s)	0.08s (0.02s)	0.71s (0.18s)	0.77s (0.11s)	0.08s (0.02s)	0.83s (0.23s)	0.73s (0.10s)	0.08s (0.02s)	0.96s (0.25s)	0.70s (0.08s)

SPC: Spectral Clustering / LPA: Label Propagation Method / LM: LabelMod

Both Table 3.1 and Table 3.2 show that the standard deviation of LabelMod is much smaller than the rest. The algorithm produces stable clustering results. These results also imply that LabelMod will still produce a solid clustering result whether the network is sparse or dense. This stability is one of LabelMod's advantages that makes the algorithm fascinating even with a slower performance time. With the same graph, the label propagation method produces stochastic results. Raghavan et al., who proposed this algorithm also suggest using an aggregation method to deal with the stochastic results [57]. Although the aggregation method is not complex, the user should run it several times before aggregating which requires multiple computational times.

It is true that random results can sometimes be better than deterministic results. However, the performance time is one of the most important issues since graph clustering is more like a pre-processing step for further analysis. Therefore, algorithms cannot be tested an infinite number of times, and deterministic results are preferred.

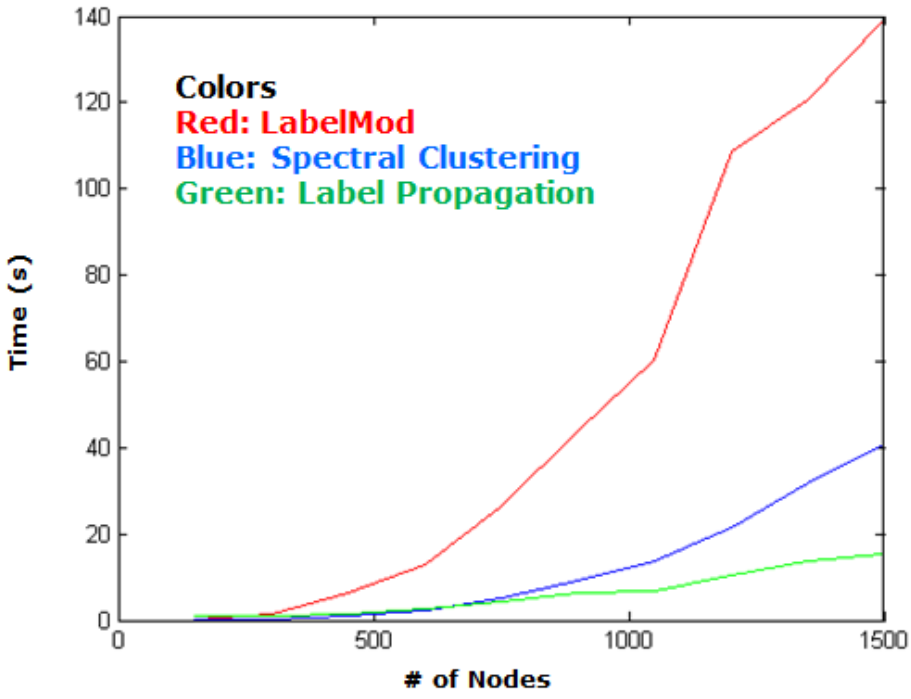


Figure 3.6. Performance Time Comparison.

In addition, algorithms were tested with a larger number of nodes. Performance time was measured using MATLAB with Intel(R) Core (TM) i5-3337U CPU @ 1.80GHz processor that has 4 GB of RAM. Figure 3.6 is a plot comparing a number of nodes to the performance time. Both LabelMod's result and spectral clustering's result show exponential shapes of a curve; however, LabelMod's performance time was worse. The gap of performance time is due to the coding that is not optimized. The label propagation method shows a linear line because the algorithm is not affected by the number of the nodes. The computational complexity of LabelMod [74, 75] and spectral clustering [76] are $O(n^2)$ while the label propagation method is $O(m)$ where m is the number of total edges [57]. The label propagation method is expected to have better performance in a sparse network. However, the algorithm will have a significant performance disadvantage in a dense network.

3.3.2 Real Graphs

In graph clustering, there are real graphs where the ground truth communities are known. In this section, LabelMod was tested in the real graphs in which the true communities are already known. The first network is called Zachary's Karate Club network[1]. In this network, nodes are 34 members of the club and edges are their friendship status [77]. This club later has a conflict between the instructor and the president which are two ground truth communities.

The second network is the US College Football network where nodes are colleges and edges are produced by Division I games for the 2000 season. In the 2000 season, there were 115 colleges in 12 conferences. In this network, there are 12 ground truth communities [14].

The last network is the Dolphins network. A network of 62 bottlenose dolphins was observed over seven years. Edges were formed when two dolphins were paired

[1]http://www.statistics.com/news/43/192/Week-14-Network-Analytics/?showtemplate=true

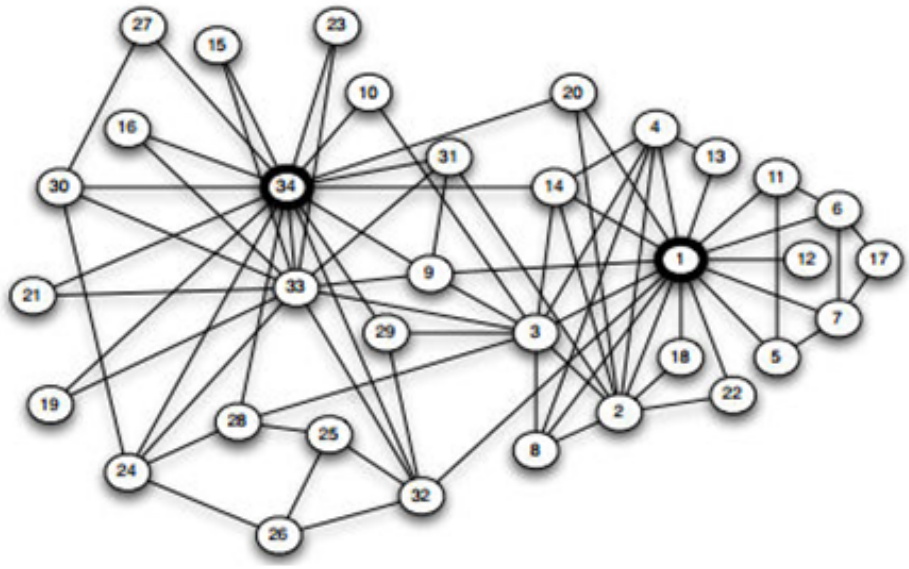


Figure 3.7. Zachary's Karate Club Network Example.

more frequently than expected. The ground truth communities of this network were the sex groups that were equal to two [78].

Table 3.3: Real Graphs Result

	Karate			**Football**			**Dolphins**		
	SPC	**LPA**	**LM**	**SPC**	**LPA**	**LM**	**SPC**	**LPA**	**LM**
RI	0.9265	0.9118	1	0.7312	0.9790	0.9790	0.5860	0.6989	0.6882
PUR	1	0.9706	1	0.3217	0.9304	0.9304	0.6613	0.7742	0.7097
NMI	0.8384	0.6861	1	0.5879	0.9269	0.9269	0.1240	0.2626	0.1391
Modularity	0.1524	0.1759	0.1796	0.1690	0.1828	0.1828	0.1647	0.1968	0.1992
Time	0.0469s	0.0625s	0.1250s	0.1406s	0.3281s	0.9844s	0.0938s	0.2813s	0.2500s

SPC: Spectral Clustering / LPA: Label Propagation Method / LM: LabelMod

The test result in Table 3.3 shows that LabelMod outperformed the other two algorithms in the Karate network. In contrast, LabelMod's performances were comparable in the Football and Dolphins networks. The Dolphins network is a real

network in which community structure is weak, and the clustering performance for all algorithms is lower compared with the other two networks. As shown in our simulated graphs, LabelMod's performance time was still higher than the other algorithms. However with a more dense network like the Dolphins network, LabelMod was faster than the label propagation methods. The performance time of LabelMod depends on the number of nodes, and a larger number of edges reduces LabelMod's performance time. Both the real graph results show that each algorithm has both strengths and weaknesses. Spectral clustering usually performs well in non-sparse networks, and it performs better when a priori information on the number of ground truth communities is given. The label propagation method performed well for a short period, but it has disadvantages of producing different results with the same network. On the other hand, LabelMod automatically detects communities without having a priori information with a stable performance. Because LabelMod uses a modularity operator, it always produces higher modularity than other algorithms that do not use a modularity optimization.

Graph clustering is a field of study where both time and accuracy are important. If nodes cluster well, it can be utilized as a useful tool for further analyses like social behavior, target marketing or bioinformatics. Many algorithms are developed, and different approaches have been made for several years. It is difficult to prove which algorithm is superior; however, LabelMod is the algorithm that yields stable results within a reasonable performance time.

4. LABELMOD IN MULTI-LAYER GRAPHS

4.1 Algorithm Development

In this chapter, we propose a new integration concept with a LabelMod algorithm that was introduced in the previous chapter. Due to LabelMod's characteristics that the algorithm can be applied to any network, it can be utilized in multi-layer graphs as well. It is common that most algorithms integrate results after finding nodes' community membership. However, our proposed algorithm aggregates graphs beforehand to make one weighted graph and process the actual clustering steps afterward. Adjacency matrices in different layers can be simply added and produce a new weighted matrix that represents all layers. Then, a label distribution matrix can be made by summing each layers probability matrix multiplied by the score of the graph. The score of the graph is described as follows:

$$s_i = \frac{E_i}{\sum_{i=1}^{G} E_i} \tag{4.1}$$

Here, E_i is the total degree of edges and G is the total number of layers. Once a weighted graph and a new propagation matrix have been created, LabelMod can be applied just like finding communities in a single layer network. The algorithm continues iterative steps of propagation operator, inflation operator, cut-off operator, and modularity operator until the stopping criterion is satisfied. The stopping criterion checks five consecutive modularity values for equality or for showing a decreasing trend.

Algorithm **LabelMod in Multi-layer Graphs**

1: **Input:**
$A_i : n \times n$ adjacency matrix of graph i.
$P_i : n \times n$ probability matrix of graph i.

2: Calculate s_i, the score of graph

3: $P_{norm} = \sum_{i=1}^{K} P_i \times s_i$

4: $W = \sum_{i=1}^{K} A_i$

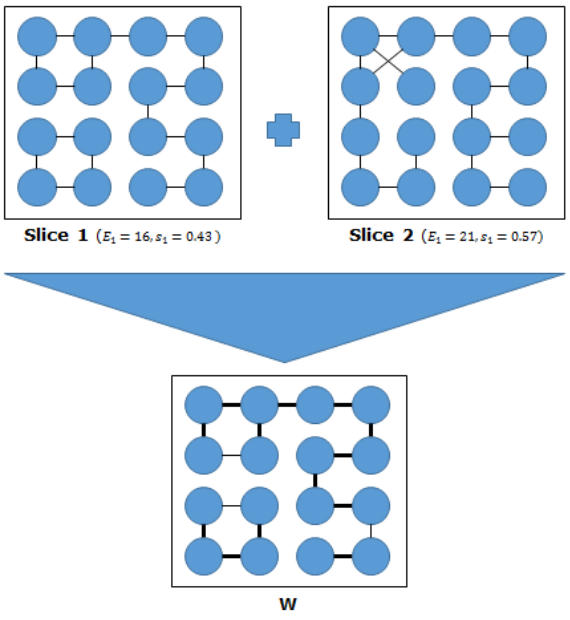
3: **Repeat**
Propagation Operator: $P_{norm}{}' = W \times P_{norm}$
Inflation Operator: $P_{norm}{}'' = \Gamma_{in} P_{norm}{}' = (P_{norm}{}')^{in} / \sum_{j=1}^{n} (P_{norm}{}')^{in}$
Cutoff Operator: Assign any value that is less than the cutoff parameter, r, to 0.
Modularity Operator: $Q = \frac{1}{4m} Tr(S^T B S)$

7: **Until:** The modularity of five consecutive iterations are equal or showing a decreasing trend.

8: Among all iterations, choose an iteration that has the maximum modularity.

9: **Output:**
$C_1, \ldots, C_k$: The cluster assignment.

(a) Pseudo Codes.



(b) Network Aggregation Example.

Figure 4.1. LabelMod Algorithm in Multi-layer Networks.

Algorithm **Spectral Clustering in Multi-layer Graphs**

1: **Input:**
W_i: $n \times n$ weighted adjacency matrices of graph layers i.
k: Target number of clusters.
α: Regularization parameter

2: Compute the normalized Laplacian matrix L_i and the subspace representation U_i for each G_i.

3: Compute the modified Laplacian matrix $L_{mod} = \sum_{i=1}^{M} L_i - \alpha \sum_{i=1}^{M} U_i U_i'$.

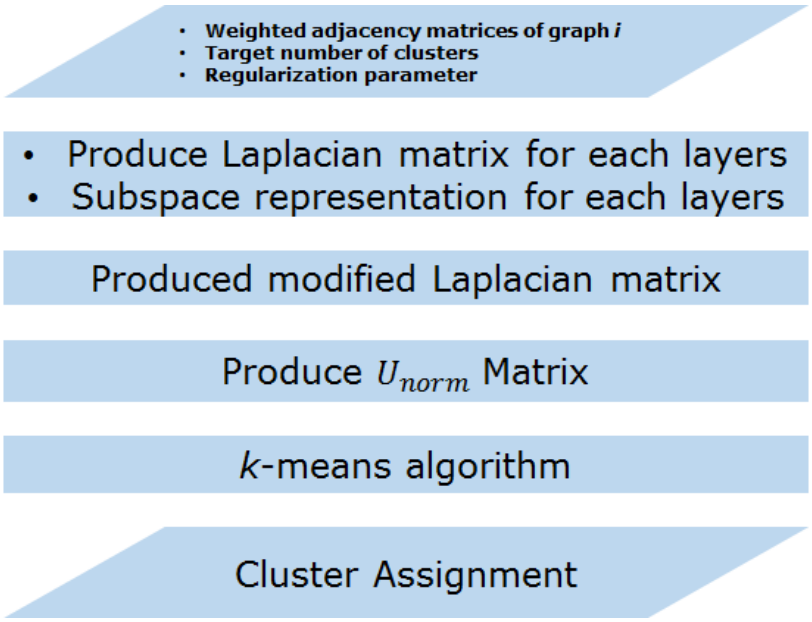
4: Compute $U \in \mathbb{R}^{n \times k}$ that is the matrix containing the first k eigenvectors $u_1, \ldots, u_k$ of L_{mod}. Normalize each row of U to get U_{norm}.

5: Let $y_i \in \mathbb{R}^k$ $(j = 1, \ldots, n)$ be the j-th row of U.

6: Cluster $y_i \in \mathbb{R}^k$ into k clusters $C_1, \ldots, C_k$ using the k-means algorithm.

7: **Output:**
$C_1, \ldots, C_k$: The cluster assignment.

(a) Pseudo Codes.



(b) Flowchart.

Figure 4.2. Spectral Clustering Algorithm in Multi-layer Networks.

4.2 Results

The algorithm was tested with experimental data that are similar to those in the previous chapter and synthetic gene co-expression data for performance comparison. The performance evaluation methods are the random index, purity, and normalized mutual information. Modularity was also compared to see if LabelMod's modularity was higher than the comparison algorithm. The spectral clustering algorithm in multi-layer graphs [79] uses a modified Laplacian matrix instead of a regular Laplacian matrix by eigenvector decomposition. The rest of the algorithm follows similar rules of the original spectral clustering algorithm. Parameter α plays a significant role in the clustering results. Pseudo code for the spectral clustering algorithm in multi-layer graphs is provided above:

4.2.1 Graph Simulation

By using a Stochastic Block Model (SBM), an adjacency matrix was created based on four different probability matrices as in Chapter 3. Then this procedure was performed with 50 runs having three communities in each run. When the network was produced with equal community sizes, the network contained 50 nodes in each community. In the second condition where the network was made with different community sizes, two communities had 30 nodes, and one community had 90 nodes. Three layers of networks were produced.

Table 4.1: Simulated Graphs Result (Same Community Size)

	P_1		P_2		P_3		P_4	
	SPC	**LM**	**SPC**	**LM**	**SPC**	**LM**	**SPC**	**LM**
RI	0.90 (0.13)	1 (0)	0.95 (0.11)	1 (0)	0.93 (0.12)	1 (0)	0.95 (0.11)	1 (0)
PUR	0.88 (0.16)	1 (0)	0.93 (0.13)	1 (0)	0.91 (0.15)	1 (0)	0.93 (0.13)	1 (0)
NMI	0.87 (0.17)	1 (0)	0.93 (0.14)	1 (0)	0.91 (0.15)	1 (0)	0.93 (0.14)	1 (0)
Modularity	0.2653 (0.06)	0.3079 (0)	0.2651 (0.05)	0.2872 (0)	0.2421 (0.05)	0.2683 (0)	0.2318 (0.04)	0.2510 (0)
Time (s)	0.14 (0.03)	0.28 (0.05)	0.19 (0.03)	0.29 (0.05)	0.21 (0.03)	0.32 (0.06)	0.21 (0.03)	0.32 (0.06)

SPC: Spectral Clustering / LM: LabelMod

Table 4.2: Simulated Graphs Result (Different Community Size)

	P_1		P_2		P_3		P_4	
	SPC	**LM**	**SPC**	**LM**	**SPC**	**LM**	**SPC**	**LM**
RI	0.89 (0.12)	1 (0)	0.91 (0.12)	0.99 (0.02)	0.95 (0.10)	0.96 (0.04)	0.95 (0.10)	0.94 (0.04)
PUR	0.91 (0.10)	1 (0)	0.93 (0.09)	1 (0)	0.95 (0.08)	0.99 (0.02)	0.96 (0.08)	0.98 (0.01)
NMI	0.85 (0.17)	1 (0)	0.88 (0.16)	0.99 (0.05)	0.91 (0.08)	0.87 (0.12)	0.92 (0.13)	0.78 (0.10)
Modularity	0.1568 (0.03)	0.1780 (0)	0.1538 (0.03)	0.1705 (0)	0.1558 (0.02)	0.1509 (0.02)	0.1519 (0.02)	0.1368 (0.01)
Time (s)	0.16 (0.03)	0.37 (0.08)	0.20 (0.03)	0.34 (0.07)	0.21 (0.03)	0.30 (0.08)	0.20 (0.03)	0.24 (0.04)

SPC: Spectral Clustering / LM: LabelMod

When the community size was the same, LabelMod found a perfect clustering for all networks. All random index, purity, and normalized mutual information were 1. From this result, we found that LabelMod produces more accurate results when networks are aggregated together. Modularity was also higher than the spectral clustering algorithm in all networks. On the other hand, the spectral clustering algorithm also produced strong performance measures, but the standard deviations for all measures were high. Most of its results formed an accurate clustering, but a poor clustering was occasionally made. The result has once again showed that LabelMod is an algorithm that produces stable clustering results. The performance time was still higher than spectral clustering's performance time. However, the performance time gap between spectral clustering and LabelMod decreased. Since the spectral clustering algorithm in multi-layer graphs computes a Laplacian matrix of all layers, the performance time of LabelMod is much shorter than spectral clustering in a condition where numerous layers of graphs exist.

When the network had different community size, LabelMod's performance did not outperform spectral clustering as in the experiment where the community size was the same. However, the performance was competitive with spectral clustering's result. Spectral clustering did not perform well in the P_1 network because this P_1 network does not produce any between edges. LabelMod's modularity value was always higher than spectral clustering in Table 4.1. However, spectral clusterings' modularity values were higher in P_3 and P_4 in Table 4.2.

Performance time was measured according to the total node sizes in networks with MATLAB. The performance time was tested with an Intel(R) Core (TM) i5-3337U CPU @ 1.80GHz processor that has a 4 GB of RAM. The test was conducted to see how LabelMod performs in an extensive network. In Figure 4.3, performance time for both algorithms is similar to the exponential distribution. However, LabelMod has a more acute angle of the slope.

The computational complexity of both algorithms is $O_(n^2)$ which is due to the $n \times n$ matrix multiplication. Since LabelMod's codes were not optimized using MATLAB,

the time gap still exist. However, LabelMod's performance time can be potentially improved by coding with a sparse matrix function.

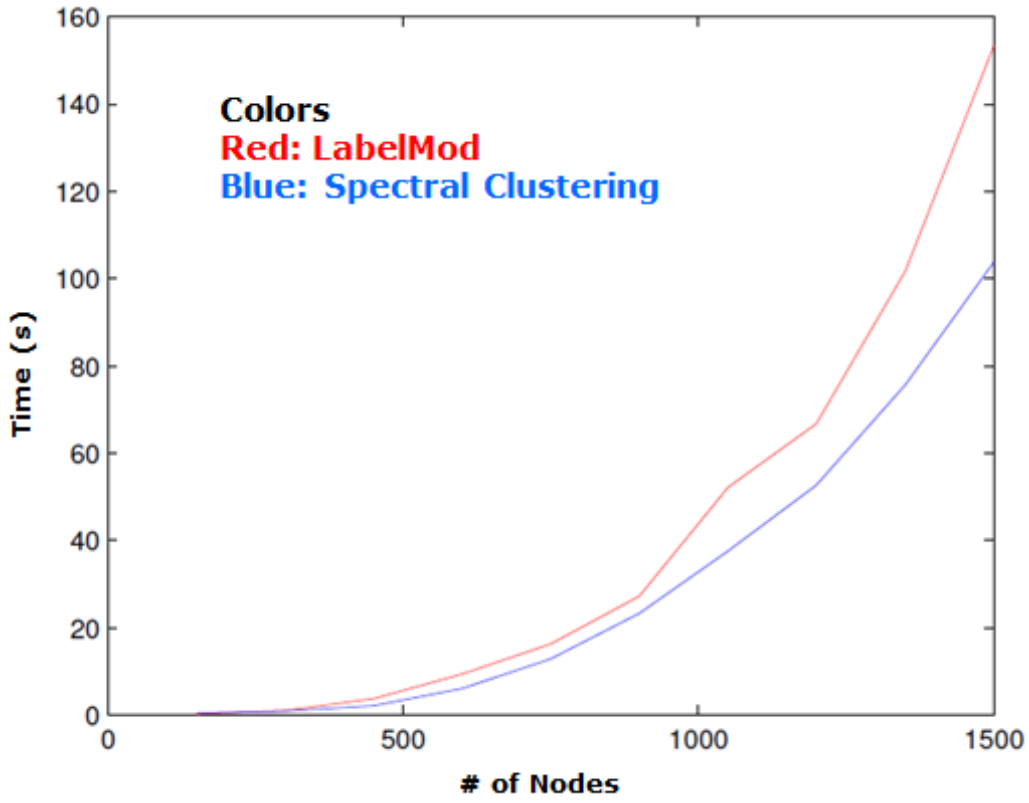


Figure 4.3. Performance Time Comparison

4.2.2 Gene Co-expression Graphs

A synthetic gene co-expression dataset was utilized to check the validation of LabelMod's performances in multi-layer networks as in the previous research [59, 80]. The network was simulated with 50 conditions, and each gene was assigned to have one of 15 ground truth communities. Five samples were chosen, and three layers were included in each sample. A detailed description of constructing a gene co-expression network is as follows:

1. Pre-assign each node to one of 15 ground truth communities.

2. Compute Pearson's correlation coefficient (Equation 4.2) with 50 conditions by each node.

3. If the absolute value of the Pearson's correlation coefficient is larger than a threshold parameter α, connect two nodes with an edge.

4. With given edge information, produce an adjacency matrix.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.2}$$

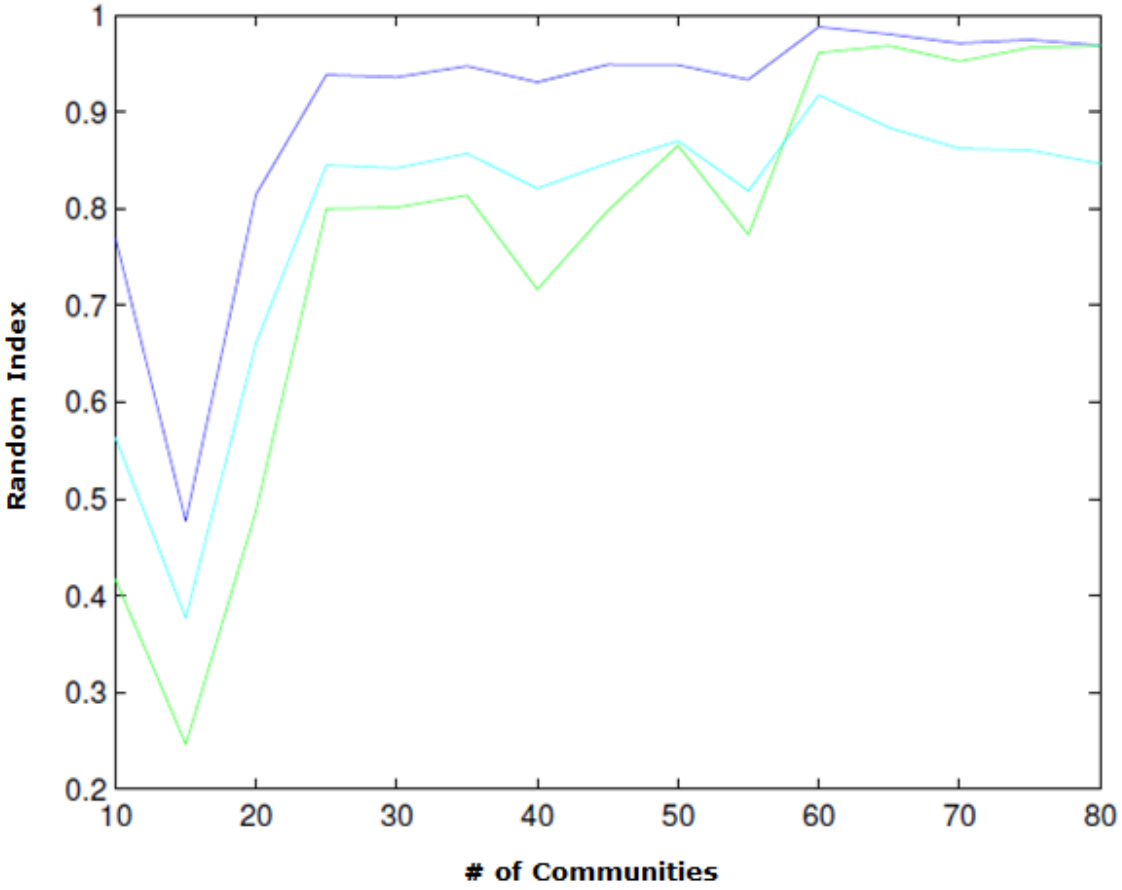
Table 4.3: Gene Co-expression Data Result

	data0		**data1**		**data2**		**data3**		**data4**	
	SPC	**LM**	**SPC**	**LM**	**SPC**	**LM**	**SPC**	**LM**	**SPC**	**LM**
RI	0.9746	0.9866	0.9663	0.9850	0.9811	0.9916	0.9585	0.9691	0.9856	0.9772
PUR	0.9663	0.9858	0.9383	0.9854	0.9940	0.9985	0.9937	1.0000	0.9752	0.8954
NMI	0.8620	0.8865	0.7980	0.8632	0.8784	0.9182	0.7591	0.7688	0.8976	0.9095
Modularity	0.4449	0.5136	0.6146	0.8082	0.5985	0.7738	0.1852	0.2160	0.6434	0.7457
Time (s)	6.3438	5.3594	8.7969	27.2031	11.2500	11.5000	17.4219	10.3906	5.9531	6.8125

SPC: Spectral Clustering / LM: LabelMod

In Table 4.3, random index, purity, normalized mutual information, modularity, and performance time were measured with five samples. In spectral clustering, some desired community should be pre-assigned. Gene co-expression data have networks where many isolated nodes exist. Isolated nodes are nodes that itself forms a community at the end of the algorithm. In this type of network, selecting a ground truth community number will not be a good choice. If a ground truth community number is used in the spectral clustering, the k-means clustering might not work well and may result in poor performance values. Figure 4.4 shows that modularity suddenly drops at k=15. It will be the best practice to avoid 15 as a parameter value. Instead

of using a ground truth community number in our study, we used the number of communities that are found in LabelMod as a parameter value.



Different color indicates different networks.

Figure 4.4. Random Index by the Selection of Community Number

LabelMod's performance measures were relatively high that the random index range from 0.9691 to 0.9916, purity range from 0.8954 to 1, normalized mutual information range from 0.7688 to 0.9182, and modularity range from 0.2160 to 0.8082. Except for data3, the four other samples' results were similar. Regarding the normalized mutual information, LabelMod performed better than the spectral clustering.

Performance time was quite different from previous studies. The result shows that LabelMod had a lower performance time in data0 and data3. In most of the

networks, the performance time was competitive except for data1. The performance time indicates that LabelMod applied in multi-layer networks can be beneficial regarding the performance time, since layers are combined before the main process of the algorithm.

By simply aggregating the layers' adjacency matrices, one weighted adjacency matrix is produced. In LabelMod, a weighted adjacency matrix and a normalized propagation matrix were used as inputs. Figures 4.5 to 4.8 show that a mere aggregation can be informative. Black lines are individual layer's results, and the red line is LabelMod's result after the aggregation. The blue asterisk is the performance measure output by LabelMod. The figures below show that the gap between a single layer's result performance curves and the aggregated result performance curves is large. The gap clearly indicates that LabelMod is an algorithm that performs well in the multi-layer networks.

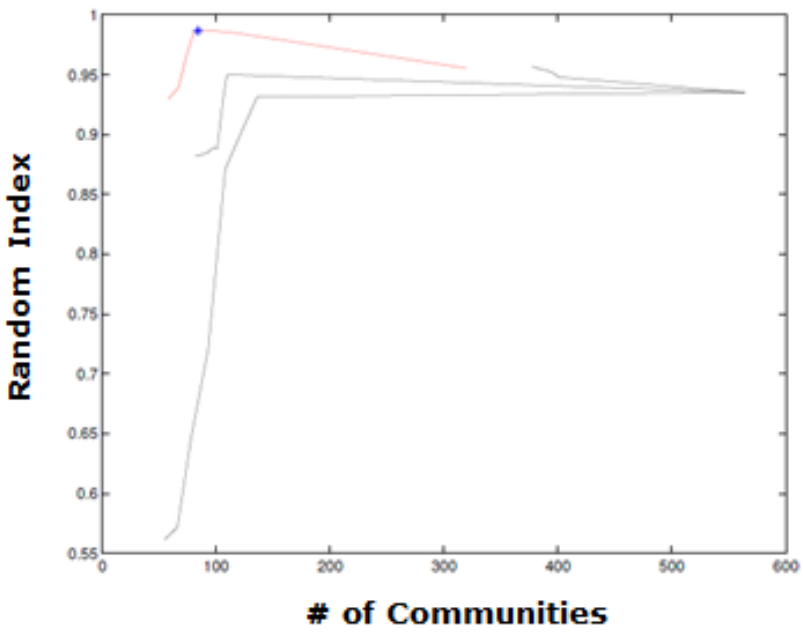


Figure 4.5. Random Index of Individual Layers and Layers Combined.

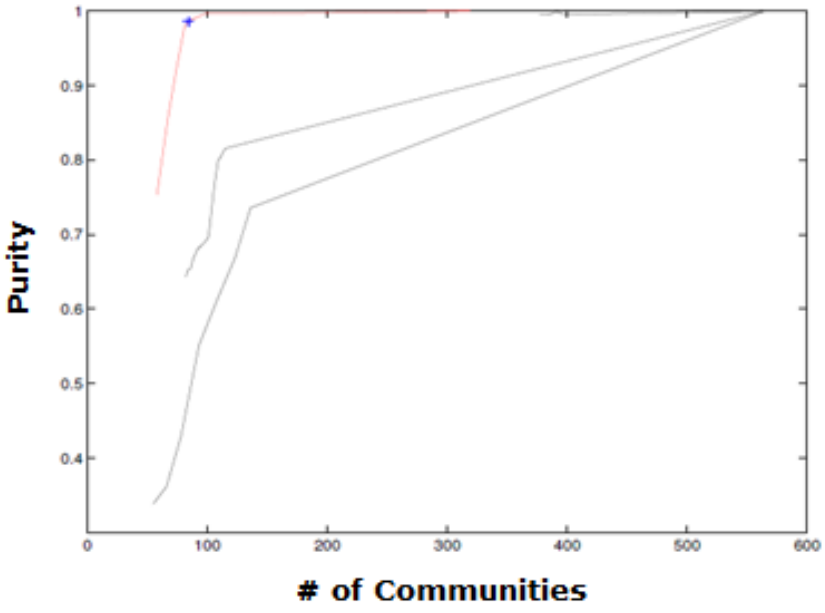


Figure 4.6. Purity of Individual Layers and Layers Combined.

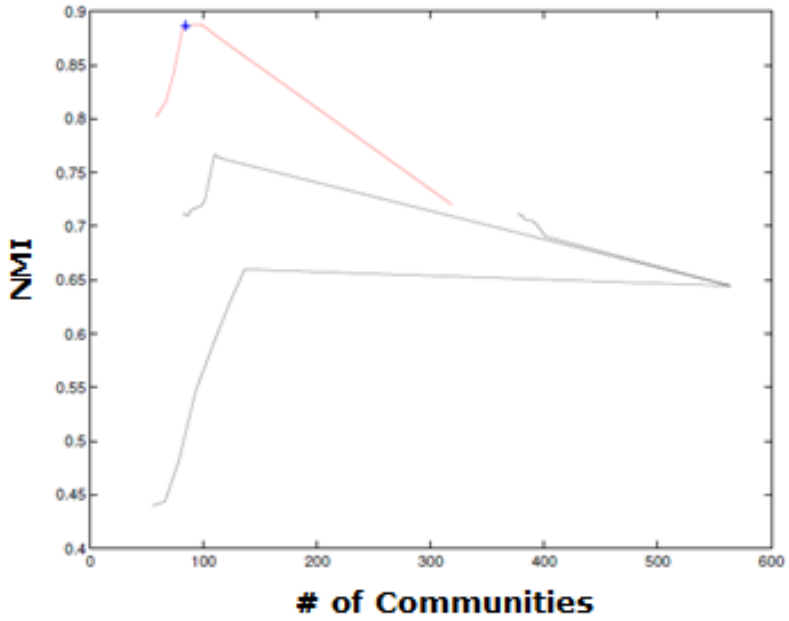


Figure 4.7. Normalized Mutual Information of Individual Layers and Layers Combined.

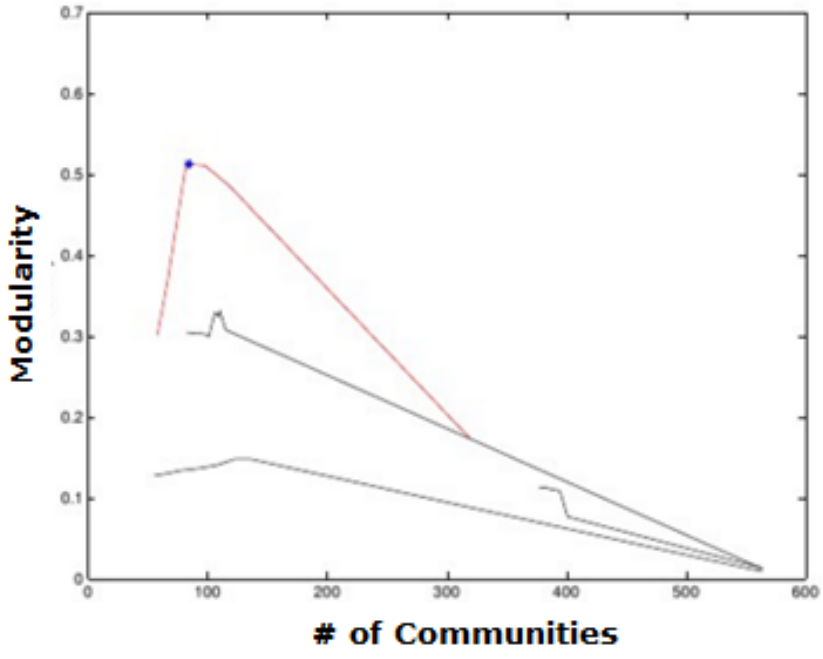


Figure 4.8. Modularity of Individual Layers and Layers Combined.

The following figure shows all random index, purity, normalized mutual information and modularity together. The blue line is a random index, the green line is purity, the black line is normalized mutual information, and the red line is modularity. A blue asterisk indicates the modularity selected by LabelMod. Random index, purity, and normalized mutual information are maximized when the modularity is at the highest point. All four curves have similar shapes so that all performance measures increase when a number of communities increase and then they all suddenly decrease after the maximum has been reached. The modularity curve is especially similar to the normalized mutual information curve. In previous studies, higher modularity also produced higher normalized mutual information.

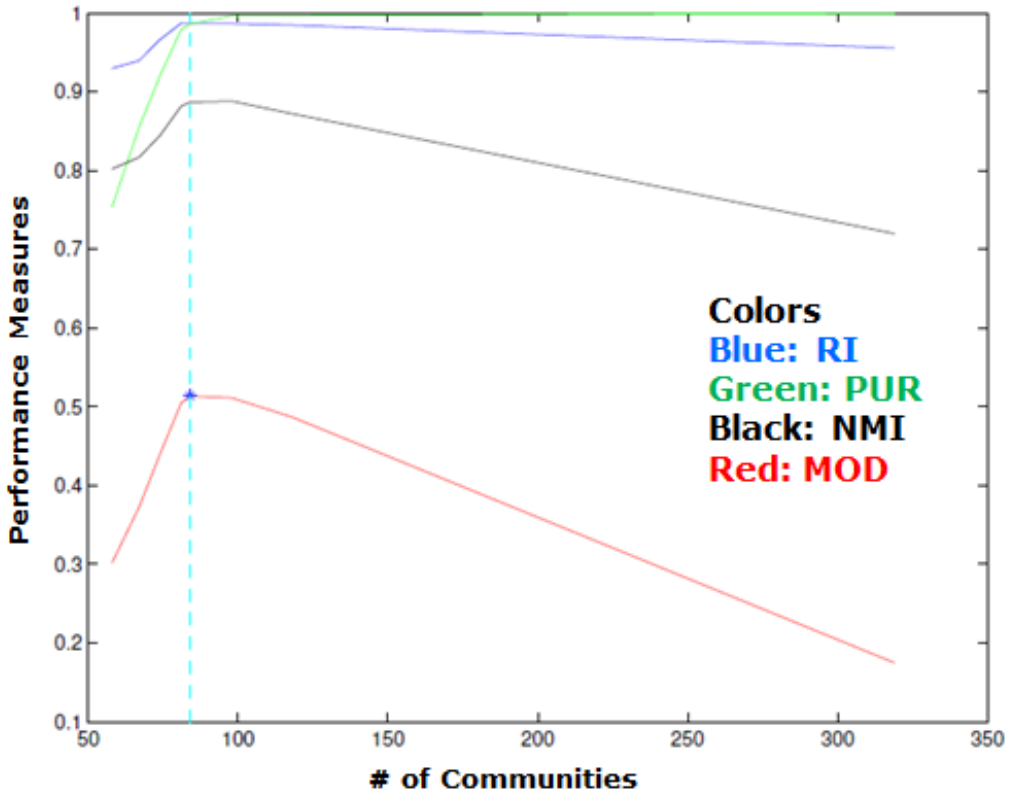


Figure 4.9. RI, PUR, NMI, and MOD in One of Individual Layers and Layers Combined.

In LabelMod, two parameters need to be assigned. The first parameter is r where it removes all values that are less than r in the cut-off operator. In Figure 4.10, modularities were computed with different r values in data0 of the gene co-expression network. When r was greater than 0.017, modularity suddenly dropped. In this case, a probability matrix value that is greater than 0.017 has a meaning. If r is greater than 0.017, the algorithm might eliminate significant probability and will result in a poor clustering. Therefore, selecting a small r will maximize the outcome. However, in a large complex network, the selection of r should be more careful. Too small an r value can increase the number of iterations in a large complex network. More studies should be conducted to find a good range of r.

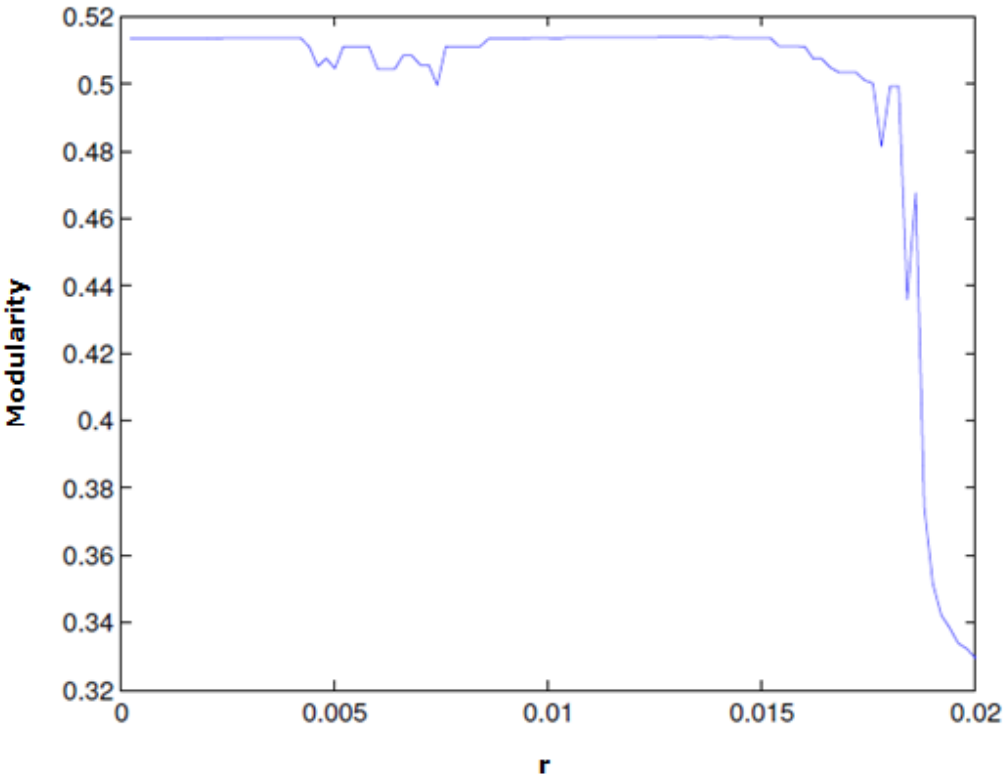


Figure 4.10. Modularity with Different r Values.

The second parameter is *in* where it is used in the inflation operator. As shown in Figure 4.11 and Figure 4.12 as the inflation parameter *in* increases, both the

modularity and normalized mutual information curves tend to form an S-shaped line for data0 of the gene co-expression network. When *in* is greater than a certain point, both modularity and normalized mutual information values become random. In our case, the threshold was around 150. In Figure 4.11, we found that LabelMod detected a smaller number of communities when *in* was significant and higher *in* produced a lower normalized mutual information as in Figure 4.12.

Parameter setting solely depends on the user; it can be determined from previous knowledge or even intuitively. However, a guideline for deciding which parameter to select is important. We found that a small number like 2 produced the best performance. Although a large *in* value might decrease the number of iterations in the algorithm, it will take longer since the matrix has to normalize with a significant degree of power. Indeed, a large *in* value might result in lower normalized mutual information.

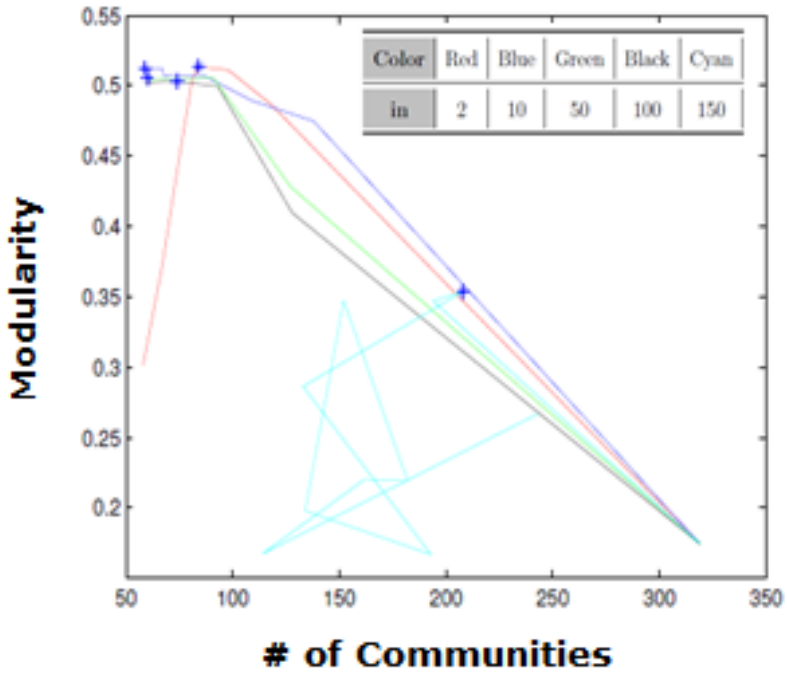


Figure 4.11. Modularity by Changing Parameter *in*.

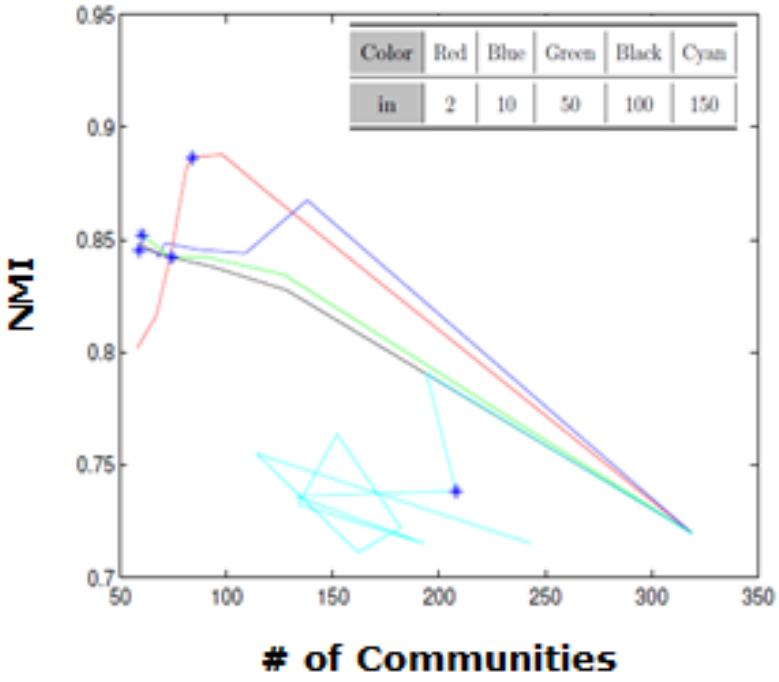


Figure 4.12. Normalized Mutual Information by Changing Parameter *in*.

In this thesis, another effort was made to implement the G-test as a stopping criterion. The G-test is a maximum likelihood statistical significance test that can be a possible substitute of the modularity operator. We computed negative log p-values of the G test whose null distribution is given by the chi-square distribution. However, this approach was not successful and was subsequently disregarded.

In this chapter, LabelMod was applied to both simulated graphs and synthetic gene co-expression data. Performances of LabelMod were similar or slightly better than the performance of spectral clustering. LabelMod's performance time was higher than spectral clustering's time due to an additional step of computing the modularity value in each iteration. However, both algorithms have a computational complexity of $O(n^2)$ and are expected to have no difference if LabelMod's codes are optimized. However, LabelMod's performance time can outperform spectral clustering's time in the condition where there are more layers of networks. LabelMod in multi-layer graphs simply combines layers of networks into one graph before iterative steps of

the algorithm. However, our study shows that even with a simple network aggregation, the algorithm produced an excellent result and showed that it is powerful. In multi-layer networks where ground truth community information is unknown, LabelMod will produce one result with a high modularity value and normalized mutual information.

5. CONCLUSIONS AND FUTURE RESEARCH

With the development of computer technologies, we are living in the era of big data where information is overwhelming. The amount of data produced is greater than we can store and analyze. When the information is overwhelming, defining a community structure can be a good start. By using algorithms, graph clustering can detect similar vertices in a network and help to reveal characteristics of communities in further analysis.

Graph clustering is a recent research field in which the concept first became of interest approximately a decade ago. After the emergence of graph clustering, many algorithms were developed in different fields. Fortunato mentions in his paper that graph clustering requires a theoretical framework specifying what the graph clustering should do [1]. Acknowledging the fact that graph clustering still needs a framework, one of our research objectives was to develop an algorithm to reveal new findings in graph clustering and contribute to future research.

5.1 Contributions

The following are the main contributions of this research:

1. **Development of LabelRank**: In this thesis, we proposed a new clustering algorithm by replacing the conditional update operator in LabelRank with the modularity operator. Most algorithms have difficulty performing well within a short performance time. We found that computing modularity in each iteration is an additional task and increases its computational complexity. However, by including the modularity operator, the new algorithm produces more accurate clustering results.

2. **Performance stability**: We expect that our algorithm can perform on any network. We developed a general clustering algorithm that can be used as a pre-processing step for further analyses. The algorithm is deterministic and produces the same clustering result. One of the advantages of our proposed algorithm is that it does not require any prior information for selecting community numbers.

3. **Aggregation of networks**: In this research, layers of networks were aggregated to create one weighted graph. A normalized probability matrix was also computed by summing each layer's probability matrix and the score function. It was shown that a simple aggregation can still be powerful.

4. **Parameter settings and effects**: Our proposed algorithm has two parameters. Different parameter settings were tested, and we found that setting a cut-off parameter to 0.001 and an inflation parameter to 2 produced the best performance. We also found that when the cut-off parameter increases and reaches a certain point, the clustering fails because it removes meaningful values in the propagation matrix. Compared with the cut-off parameter, the inflation parameter plays a more important role. As the inflation parameter increases, both the modularity curve and the normalized mutual information curve forms S-shaped lines. With a larger inflation parameter, a smaller number of communities were detected. However, when the inflation parameter reaches a certain point, both modularity and normalized mutual information produced random curves.

5. **Modularity and normalized mutual information**: We found that the modularity computed by LabelMod follows a similar pattern with its normalized mutual information value. Modularity is correlated with normalized mutual information and LabelMod will perform well regarding normalized mutual information value.

5.2 Future Research

The following are possible future research topics:

1. **Modularity and accuracy measures**: Although modularity is known as the most common quality function in graph clustering, high modularity does not always guarantee high accuracy in performance. The nature of modularity needs to be studied more and a guideline should be provided about why a high modularity occasionally fails to have high accurate performance. Then a new quality function that evaluates clustering performance can be developed.

2. **Developing a more efficient algorithm with a new stopping criterion**: The current algorithm's stopping criterion is too intuitive. Instead of using an intuitive approach, a new stopping criterion using a more statistical approach is suggested. A statistical stopping criterion using mutual information will likely produce better clustering.

3. **Method of network aggregation**: We aggregated networks by simply summing the layers; in contrast, a more mathematical approach to network aggregation will be helpful. If layers of networks can be successfully aggregated into one network, any single-layer algorithm can be applied. Then, graph clustering in multi-layer networks becomes easier.

4. **Overlapping nature of community in multi-layer networks**: There are nodes that can have more than one community assignment. By using the characteristics of modularity, the algorithm can be developed when there are nodes that have multiple community assignments.

5. **Framework on clustering algorithm, benchmark graph**: In graph clustering, deciding which algorithm is better remains controversial due to a lack of the fixed definition. Benchmark graphs are needed to be developed, and a clear definition of clustering should be established.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[2] Mark E.J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

[3] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge University Press, 1994.

[4] John Scott. *Social Network Analysis*. Sage, 2012.

[5] Amanda L. Traud, Eric D. Kelsic, Peter J. Mucha, and Mason A. Porter. Community structure in online collegiate social networks. *SIAM Review*, 53(3):526–543, 2008.

[6] P. Krishna Reddy, Masaru Kitsuregawa, P. Sreekanth, and S. Srinivasa Rao. A graph based approach to extract a neighborhood customer community for collaborative filtering. In *Proceedings of Databases in Networked Information Systems*, pages 188–200. Springer, 2002.

[7] Alexander W. Rives and Timothy Galitski. Modular organization of cellular networks. *Proceedings of the National Academy of Sciences*, 100(3):1128–1133, 2003.

[8] Victor Spirin and Leonid A. Mirny. Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences*, 100(21):12123–12128, 2003.

[9] Jingchun Chen and Bo Yuan. Detecting functional modules in the yeast protein–protein interaction network. *Bioinformatics*, 22(18):2283–2290, 2006.

[10] Hawoong Jeong, Sean P. Mason, A-L Barabási, and Zoltan N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001.

[11] Eric M. Phizicky and Stanley Fields. Protein-protein interactions: methods for detection and analysis. *Microbiological reviews*, 59(1):94–123, 1995.

[12] Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and Jukka-Pekka Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *science*, 328(5980):876–878, 2010.

[13] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Petropolitanae*, 8:128–140, 1741.

[14] Michelle Girvan and Mark E.J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[15] Mark E.J. Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):321–330, 2004.

[16] Gary William Flake, Steve Lawrence, C. Lee Giles, and Frans M. Coetzee. Self-organization and identification of web communities. *Computer*, 35(3):66–70, 2002.

[17] Yon Dourisboure, Filippo Geraci, and Marco Pellegrini. Extraction and classification of dense communities in the web. In *Proceedings of the 16th International Conference on World Wide Web*, pages 461–470. ACM, 2007.

[18] Stuart L. Pimm. The structure of food webs. *Theoretical Population Biology*, 16(2):144–158, 1979.

[19] Brian W. Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970.

[20] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[21] Adam Schenker, Mark Last, Horst Bunke, and Abraham Kandel. Graph representations for web document clustering. In *Proceedings of Pattern Recognition and Image Analysis*, pages 935–942. Springer, 2003.

[22] Adel Hlaoui and Shengrui Wang. A direct approach to graph clustering. *Neural Networks and Computational Intelligence*, 4:158–163, 2004.

[23] Matthew J. Rattigan, Marc Maier, and David Jensen. Graph clustering with network structure indices. In *Proceedings of the 24th International Conference on Machine Learning*, pages 783–790. ACM, 2007.

[24] William E. Donath and Alan J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, 1973.

[25] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[26] Matthew J. Rattigan, Marc Maier, and David Jensen. Using structure indices for efficient approximation of network properties. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 357–366. ACM, 2006.

[27] Haifeng Du, Marcus W. Feldman, Shuzhuo Li, and Xiaoyi Jin. An algorithm for detecting community structure of social networks based on prior knowledge and modularity. *Complexity*, 12(3):53–60, 2007.

[28] Philipp Schuetz and Amedeo Caflisch. Multistep greedy algorithm identifies community structure in real-world and computer-generated networks. *Physical Review E*, 78(2):026112, 2008.

[29] Josep M. Pujol, Javier Béjar, and Jordi Delgado. Clustering algorithm for determining community structure in large networks. *Physical Review E*, 74(1):016107, 2006.

[30] Roger Guimera, Marta Sales-Pardo, and Luís A Nunes Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2):025101, 2004.

[31] Claire P. Massen and Jonathan P.K. Doye. Identifying communities within energy landscapes. *Physical Review E*, 71(4):046101, 2005.

[32] Andres Medus, Guillermo Acuna, and Claudio O. Dorso. Detection of community structures in networks via global optimization. *Physica A: Statistical Mechanics and its Applications*, 358(2):593–604, 2005.

[33] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 72(2):027104, 2005.

[34] Mark E.J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[35] Sune Lehmann and Lars Kai Hansen. Deterministic modularity optimization. *The European Physical Journal B*, 60(1):83–88, 2007.

[36] Andrea Capocci, Vito D.P. Servedio, Guido Caldarelli, and Francesca Colaiori. Detecting communities in large networks. *Physica A: Statistical Mechanics and its Applications*, 352(2):669–676, 2005.

[37] Fa-Yueh Wu. The potts model. *Reviews of Modern Physics*, 54(1):235–268, 1982.

[38] Marcelo Blatt, Shai Wiseman, and Eytan Domany. Superparamagnetic clustering of data. *Physical Review Letters*, 76(18):3251, 1996.

[39] Jörg Reichardt and Stefan Bornholdt. Detecting fuzzy community structures in complex networks with a potts model. *Physical Review Letters*, 93(21):218701, 2004.

[40] Barry D. Hughes. *Random Walks and random environments*. Clarendon Press, 1995.

[41] Haijun Zhou. Distance, dissimilarity index, and network community structure. *Physical Review E*, 67(6):061901, 2003.

[42] Haijun Zhou. Network landscape from a brownian particles perspective. *Physical Review E*, 67(4):041908, 2003.

[43] Stijin van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, Dutch National Research Institute for Mathematics and Computer Science, University of Utrecht, Netherlands, 2000.

[44] Mark S. Handcock, Adrian E. Raftery, and Jeremy M. Tantrum. Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2):301–354, 2007.

[45] Johan H. Koskinen and Tom A.B. Snijders. Bayesian inference for dynamic social network data. *Journal of Statistical Planning and Inference*, 137(12):3930–3938, 2007.

[46] Christopher J. Rhodes and E.M.J Keefe. Social network topology: a Bayesian approach. *Journal of the Operational Research Society*, 58(12):1605–1611, 2007.

[47] Małgorzata Rowicka and Andrzej Kudlicki. Bayesian modeling of protein interaction networks. In *Proceedings of 24th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, volume 735, pages 283–288. AIP Publishing, 2004.

[48] Johannes Berg and Michael Lässig. Cross-species analysis of biological networks by bayesian alignment. *Proceedings of the National Academy of Sciences*, 103(29):10967–10972, 2006.

[49] Francoís Lorrain and Harrison C. White. Structural equivalence of individuals in social networks. *The Journal of Mathematical Sociology*, 1(1):49–80, 1971.

[50] Douglas R. White and Karl P. Reitz. Graph and semigroup homomorphisms on networks of relations. *Social Networks*, 5(2):193–234, 1983.

[51] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[52] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[53] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[54] David J.C. MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

[55] James P. Bagrow and Erik M. Bollt. Local method for detecting communities. *Physical Review E*, 72(4):046108, 2005.

[56] Symeon Papadopoulos, Andre Skusa, Athena Vakali, Yiannis Kompatsiaris, and Nadine Wagner. Bridge bounding: A local approach for efficient community discovery in complex networks. Technical report, Informatics & Telematics Institute (CERTH), 2009.

[57] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007.

[58] Kevin Lewis, Jason Kaufman, Marco Gonzalez, Andreas Wimmer, and Nicholas Christakis. Tastes, ties, and time: A new social network dataset using facebook. com. *Social Networks*, 30(4):330–342, 2008.

[59] Shuqin Zhang, Hongyu Zhao, and Michael K. Ng. Functional module analysis for gene coexpression networks with network integration. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 12(5):1146–1160, 2015.

[60] Huiyi Hu, Yves van Gennip, Blake Hunter, Mason A. Porter, and Andrea L. Bertozzi. Multislice modularity optimization in community detection and image segmentation. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining Workshops*, pages 934–936, 2012.

[61] Duc Truong Pham, Stefan S. Dimov, and Chi D. Nguyen. Selection of K in K-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1):103–119, 2005.

[62] Steve Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10):103018, 2010.

[63] Jierui Xie, Boleslaw K. Szymanski, and Xiaoming Liu. Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *Proceedings of Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 344–349. IEEE, 2011.

[64] Jierui Xie and Boleslaw K. Szymanski. Towards linear time overlapping community detection in social networks. In *Proceedings of Advances in Knowledge Discovery and Data Mining*, pages 25–36. 2012.

[65] Jierui Xie and Boleslaw K. Szymanski. Community detection using a neighborhood strength driven label propagation algorithm. In *Proceedings of IEEE 1st International Workshop on Network Science*, pages 22–24, 2011.

[66] Gergely Tibély and Jnos Kertész. On the equivalence of the label propagation method of community detection and a potts model approach. *Physica A: Statistical Mechanics and its Applications*, 387(19):4982–4984, 2008.

[67] Jierui Xie and Boleslaw K. Szymanski. Labelrank: A stabilized label propagation algorithm for community detection in networks. In *Proceedings of 2013 IEEE 2nd Network Science Workshop (NSW)*, pages 138–143. IEEE, 2013.

[68] Anton J. Enright, Stijn Van Dongen, and Christos A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, 2002.

[69] Shuqin Zhang and Hongyu Zhao. Community identification in networks with unbalanced structure. *Physical Review E*, 85(6):066114, 2012.

[70] Xiaowen Dong, Pascal Frossard, Pierre Vandergheynst, and Nikolai Nefedov. Clustering with multi-layer graphs: A spectral perspective. *IEEE Transactions on Signal Processing*, 60(11):5820–5831, 2012.

[71] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854, 2010.

[72] Tarald O. Kvalseth. Entropy and correlation: Some comments. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(17):517–519, 1987.

[73] Peter J. Bickel and Aiyou Chen. A nonparametric view of network models and newmangirvan and other modularities. *Proceedings of the National Academy of Sciences*, 106(50):21068–21073, 2009.

[74] Thomas H Cormen. *Introduction to Algorithms*. MIT press, 2009.

[75] Rodney R Howell. On asymptotic notation with multiple variables. Technical report, Citeseer, 2008.

[76] Jialu Liu, Chi Wang, Marina Danilevsky, and Jiawei Han. Large-scale spectral clustering on graphs. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1486–1492. AAAI Press, 2013.

[77] Wayne W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[78] David Lusseau, Karsten Schneider, Oliver J. Boisseau, Patti Haase, Elisabeth Slooten, and Steve M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[79] Xiaowen Dong, Pascal Frossard, Pierre Vandergheynst, and Nikolai Nefedov. Clustering on multi-layer graphs via subspace analysis on grassmann manifolds. *IEEE Transactions on Signal Processing*, 62(4):905–918, 2014.

[80] Anbupalam Thalamuthu, Indranil Mukhopadhyay, Xiaojing Zheng, and George C. Tseng. Clustering on multi-layer graphs via subspace analysis on grassmann manifolds. *IEEE Transactions on Signal Processing*, 62(4):905–918, 2014.